

/DETERMINATION OF THE BAUD RATE OF AN FSK SIGNAL
USING ADAPTIVE NOISE CANCELLING TECHNIQUES/

by

MARC DAVID BRACK

B.S., Kansas State University, 1984

A MASTER'S THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY.

Manhattan, Kansas

1985

Approved by:

Ronald R. Hummel
Major Professor

LD
2668
.T4
1985
B72
C.2

Acknowledgements

ALL202 942380

I would like to express my thanks to the members of my advisory committee, Dr. John E. Boyer, Dr. Stephen A. Dyer, and special thanks to Dr. Donald R. Hummels, my major professor.

I would also like to thank those who helped me during the course of my work. Those people are Bob Schneider and Phil Buckland who provided support of the computing facilities and Christy Schrollier who helped with the manuscript preparation.

The financial support granted by Motorola, Inc., Government Electronics Group and by Kanaas State Univeraity is also appreciated.

Finelly, and most importantly, I would like to thank my wife, Lisa. Besides providing encouragement and support through the entire effort, she also constructed the diagrams of the manuscript and did the final proofreading.

Table of Contents

I.	Introduction	1
II.	FSK Band Rate Analysis System	4
	Bandpass Filter	4
	Limiter	4
	Delay-Line Discriminator	6
	Low-Pass Filter	10
	Adaptive Noise Canceller	10
	Zero-Crossing Estimator	18
III.	Development of a System Model	19
	Low-Pass Techniques	19
	Bandpass Pre-Filter	21
	Limiter	22
	Delay-Line Discriminator	22
	Low-Pass Post-Filter	25
	Adaptive Noise Canceller	26
	Zero-Crossing Algorithm	31
IV.	Description of the Simulation	35
	FSK Signal Modulator	35
	Noise Model	37
V.	Preliminary Observations and Estimator Definitions	43
	DLD Output	44
	ANC Investigation	47
	Estimator Definitions	62
VI.	Results and Conclusions	67
	Evaluation of Estimators	68
	Demodulator Evaluation	73
	ANC Parameter Evaluation	73
	Varying Frequency Deviation	78
	Different Baud Rates	81
	Evaluation Against the Traditional Method	85
	Conclusions	91
	Appendix A: Computer Program Listings	95
	References	162

List of Figures

	page
1. System for determining FSK baud rate	2
2. Block diagram of an FSK demodulator	5
3. Block diagram of a delay-line discriminator	7
4. Comparator output of delay-line discriminator	9
5. Adaptive noise canceller (ANC)	11
6. Effect of post-filter and limiter on the correlation properties of the DLD output	15
7. The adaptive noise canceller as a self-tuner	17
8. FSK demodulator	20
9. Low-pass equivalent model of a delay-line discriminator	23
10. Adaptive linear combiner	
11. The LMS adaptive filter implemented with a tapped delay line	29
12. Histogram analysis of the zero-crossing estimator	33
13. Effect of post-filter and limiter on DLD output	45
14. DLD output for signals of small deviation and varying SNR	46
15. DLD output for signals of large deviation and varying SNR	48
16a. ANC output information for varying delay	51
16b. ANC output information for varying delay	
17. Baud rate estimate from the weight vector for large μ	55
18. Baud rate estimate from the weight vector for small μ	56
19. Effect on weight vector of varying delay	58
20. Modified system for determining FSK baud rate	60
21. Effect of intermediate filtering on ANC input	63

List of Fignras (cont.)

	page
22. ANC outpnt information with intermediate filtering with bandwidth of 338 Hz	64
23a. Evaluation of estimators	69
23b. Evaluation of eatimators	70
23c. Evaluation of astimatora	71
23d. Evalnation of estimatora	72
24a. Evaluation of FSK demodulator	74
24b. Evalnation of FSK damodulator	75
24c. Evaluation of FSK demodnlator	76
25a. Effect of varying fragnancy daviation of tha FSK aignal on the estimators	79
25b. Effect of varying frequancy daviation of tha FSK aignal on the baudwidth	80
26a. Effact of varying the band rate on tha XZCF estimator	82
26b. Effact of varying tha band rata ou tha MLTOI estimator	83
26c. Effect of varying the band rate on tha baudwidth estimata	84
27a. Evalnation of propoaed aatimators againat tha traditional zero-crossiug estimator	87
27b. Evaluation of propoaed estimatora against tha traditional zero-crossing estimator	88
27c. Evaluation of propoaed aatimatora against tha traditional zaro-crossing estimator	89
27d. Evalnation of propoaad astimators againat the traditional zaro-crossiing estimator	90
28. Final system for determiniing FSK baud rata	92

List of Tables

	page
1. Estimator definition	66
2. Evaluation of ANC parameters	77

I. Introduction

Free space is a commonly used transmission medium for communicating information. Information communicated as such can not be kept confidential because of the nature of the medium. Thus the task of surveillance is established; there will always be someone that wishes to receive confidential information intended for someone else. Many times a surveillance receiver is not in the optimal location for reception which results in low signal-to-noise ratios. Also, the surveillance receiver can not expect to know a priori any parameters of the transmission. Furthermore, those transmitting the information will try to keep it from getting to the wrong people. For these three reasons alone, surveillance receivers are among the most challenging receivers to design.

This research effort is focused at determining the baud rate of a message transmitted using frequency shift keying (FSK) modulation. The received signal is assumed to be FSK. This assumption has been made either by a modulation recognition algorithm or by some knowledgeable insight. Instead of relying on conventional methods which use a zero-crossing algorithm, the estimate is determined using adaptive noise cancelling techniques. Figure 1 shows the two methods of a baud rate analysis system. The crux of the problem is to evaluate the performance of the adaptive noise cancelling technique in estimating the baud rate in a low signal-to-noise ratio environment.

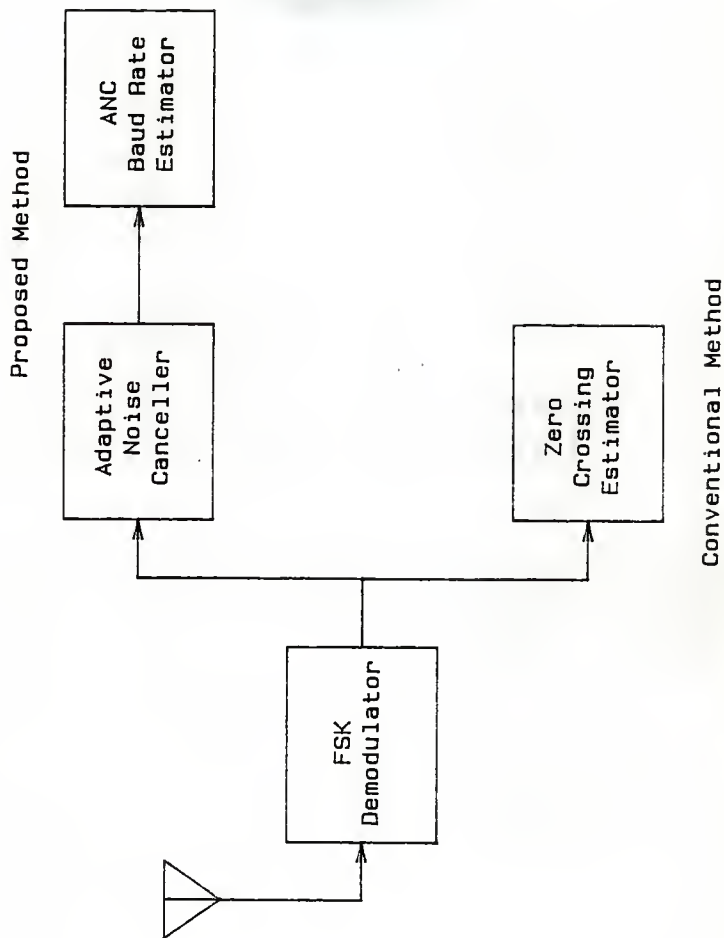


FIGURE 1: System for determining FSK baud rate.

Obviously some restrictions must be made since one receiver can not be expected to work for all signals. The first restriction is that the FSK signal bandwidth be no greater than 6000 Hz. This figure was arbitrarily chosen but can be changed to meet the needs of a specific application. The other restriction applies to the baud rate. The FSK signal bandwidth, the frequency deviation of the FSK signal, and the baud rate are roughly related by

$$B_{FSK} \cong 2(\Delta f_{FSK} + R) \quad (1.1)$$

In order to demodulate the signal with some degree of accuracy the frequency shift must be as large as the baud rate. This means that the maximum baud rate for a given signal bandwidth is one-fourth the bandwidth. For this case the main lobes of the message spectrum are above and below the center frequency. To lower the probability of error during demodulation, the frequency deviation is generally larger than the baud rate. In order that the main lobe and the first side lobe of the message spectrum be above and below the center frequency upon modulation, the frequency deviation must be at least twice the baud rate. Based on this premise, the restriction for the maximum baud rate is 1000 Hz given the 6000 Hz signal bandwidth.

Since the investigation will be done by comparing the performance of the proposed method against that of the conventional method, any benefits or harms that may result from the arbitrarily chosen restrictions will affect both methods. Therefore, final judgment will not be influenced by these restrictions.

II. FSK Band Rate Analysis System

The system used to determine the band rate of an FSK signal consists of two major devices as shown in Figure 1. The first is an FSK demodulator, the second is the band rate estimator. The band rate estimate is determined either with adaptive noise cancelling techniques, the method under investigation, or with zero-crossing techniques, the conventional method. The FSK demodulator, illustrated in Figure 2, has a delay-line discriminator (DLD) as its principal component. Before the DLD is a bandpass filter to limit the input noise power, and following the DLD is a low-pass filter for the same purpose. The limiter is to stabilize the DLD response which is dependent on the input signal amplitude.

Bandpass Filter

The pre-detection filter is a 4-pole Butterworth bandpass filter. Its purpose is twofold. First, it limits the noise power that enters the system. Second, it allows only signals for which the receiver is designed to operate to enter the system. For this reason the bandwidth is set to 6000 Hz. Also, the filter should be centered on the carrier frequency of the signal which has been determined a priori.

Limiter

FSK is a special case of frequency modulation (FM) and the FSK demodulator can actually be considered as an FM demodulator. It makes sense then to eliminate any amplitude modulation resulting from the noise perturbations which might have entered

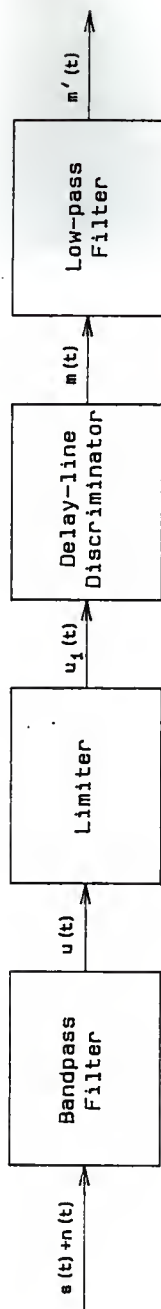


FIGURE 2: Block diagram of an FSK demodulator.

the signal. Furthermore, as will be discussed shortly, the output of the DLD for a continuous-wave input is dependent upon the squared amplitude of the input. Thus, if the input amplitude is left to vary, the DLD response will also vary.

The limiter is modeled as a device which will have an output of constant amplitude but will have identical phase properties as the input. That is, the limiter will block any amplitude modulation while frequency modulation is allowed to pass.

Delay-Line Discriminator

The DLD is a device whose output is proportional to the instantaneous frequency deviation of the input signal from its center frequency. A block diagram of the DLD is given in Figure 3. One use of the DLD is for FSK demodulation. If the input is an FSK signal the output will be the bit sequence of the binary message.

The effect of the DLD is apparent when the input consists of a noise-free unmodulated carrier, namely,

$$n_1(t) = A \cos[(2\pi f_c + 2\pi \Delta f)t] \quad (2.1)$$

The frequency offset from the carrier frequency of the signal, f_c , is denoted by Δf . It can be shown that the output of the DLD is given by

$$m(t) = -4A^2 \sin(2\pi \Delta f T + 2\pi f_c T) \quad (2.2)$$

where T is the time of delay between $n_1(t)$ and $n_2(t)$. Note this output is only for a continuous-wave input. This result plus the output of the DLD for a general input are developed with low-pass modeling techniques presented in the following section.

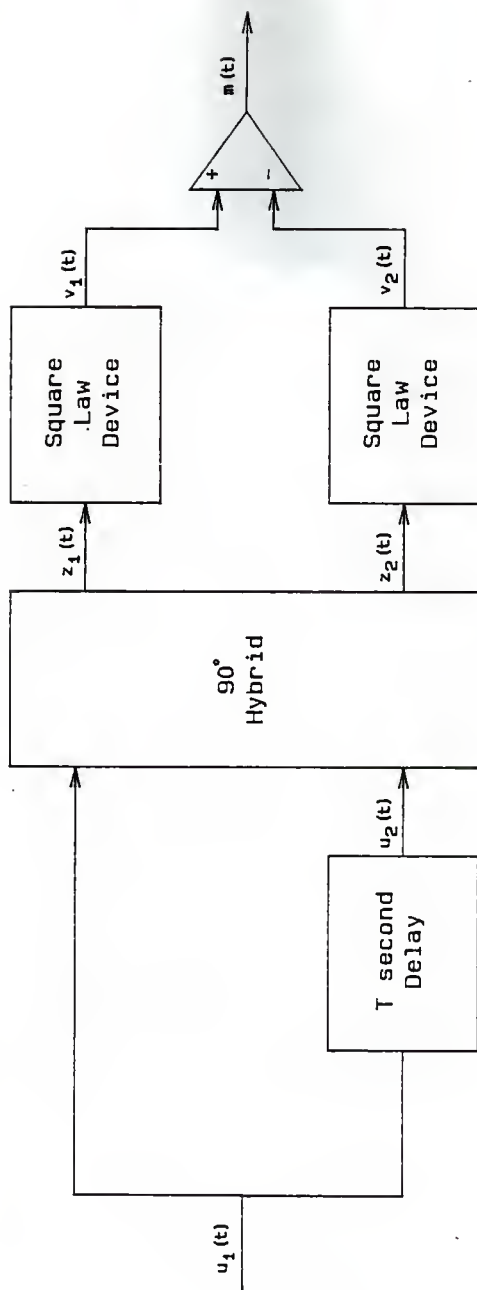


FIGURE 3: Block diagram of a delay-line discriminator.

In order for the discriminator to work properly, (i.e., produce an output proportional to instantaneous input frequency deviation) it is necessary that the output be zero when the deviation is zero. For this to be, it is necessary that

$$2\pi f_c T = n\pi \quad \text{for } n=1,2,\dots \quad (2.3)$$

For this choice, the output is described by

$$m(t) = \pm 4A^2 \sin(2\pi \Delta f T) \quad (2.4)$$

The $\pm$ comes from the addition of $n\pi$, + for n odd and - for n even. A plot of one cycle of (2.4) is shown in Figure 4 when the delay T is chosen to be 125 μsec and n is even. Observe that the curve is nearly linear when the frequency deviation is small. T should be chosen so that the discriminator operates in this linear range. The obvious reason for the choice is that it is easier to work with linear or nearly linear devices as opposed to non-linear devices. Another reason is that the response of the DLD in the non-linear region for general inputs is not understood well. Since the pre-filter has limited the range of the incoming deviations, the bandwidth of the pre-filter dictates a specific delay. The procedure in setting T is now discussed.

The linear region is roughly that portion of the spectrum centered on f_c such that the argument of the sinusoid is less than $\pi/6$ in magnitude. Then the relation between the maximum delay and the maximum deviation is

$$2\pi \Delta f_{\max} T_{\max} = \frac{\pi}{6} \quad (2.5)$$

Recall that the pre-filter passes frequencies from $-\Delta f_{\max}$ to $\Delta f_{\max}$. Substituting the pre-filter bandwidth, B_{RF} , for twice the

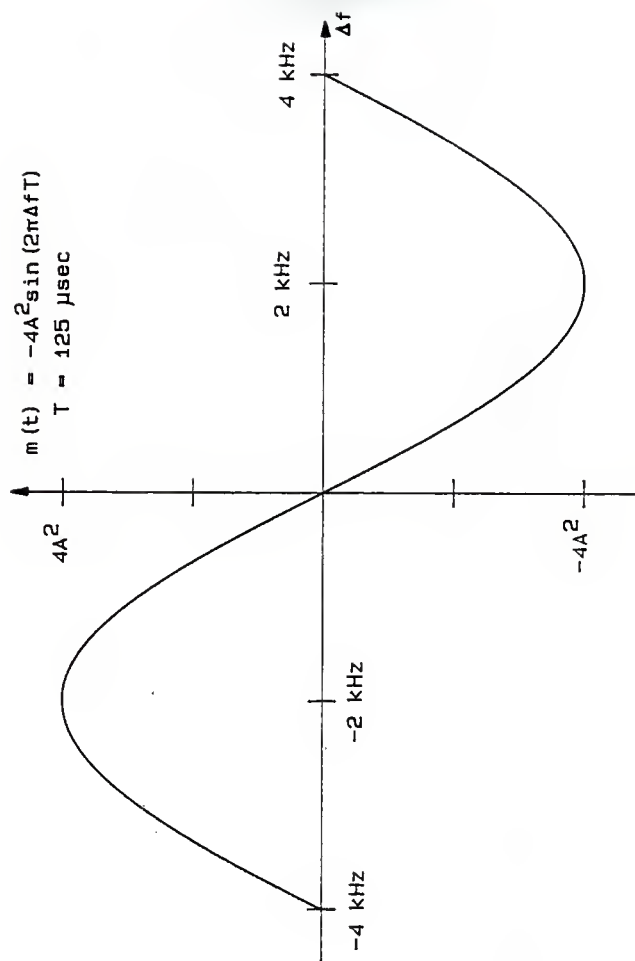


FIGURE 4: Comparator output of delay-line discriminator.

maximum deviation and then solving for the maximum delay results in

$$T_{\max} = \frac{1}{6B_{RF}} \quad (2.6)$$

Since the product $2f_c T$ must be an integer by Equ. (2.3), T should be chosen as the largest value which satisfies Eqns. (2.3) and (2.6).

Low-Pass Filter

The purpose of the post-filter is to reduce the noise in the output signal of the DLD. Since the output is at baseband, the filter is low-pass. The filter implemented is a 4-pole Butterworth filter. The bandwidth, to eliminate as much noise as possible, should be only slightly larger than the bandwidth of the output signal. However, selecting the bandwidth is difficult since the signal bandwidth is proportional to the baud rate. As stated in the introduction, the system should work for baud rates up to 1000 Hz. Thus the post-filter bandwidth is set to 1000 Hz.

Adaptive Noise Canceller

The method of baud rate analysis under study consists of an adaptive noise canceller (hereafter referred to as an ANC) and the ANC baud rate estimator. The ANC baud rate estimator is an algorithm which evaluates the parameters of the ANC to determine an estimate of the baud rate. The ANC implemented is given in Figure 5 and was introduced by Widrow [1]. The primary input to the ANC is some signal plus noise, $s+n_0$. The reference input, specified as n_1 , is also noise. It is uncorrelated with the

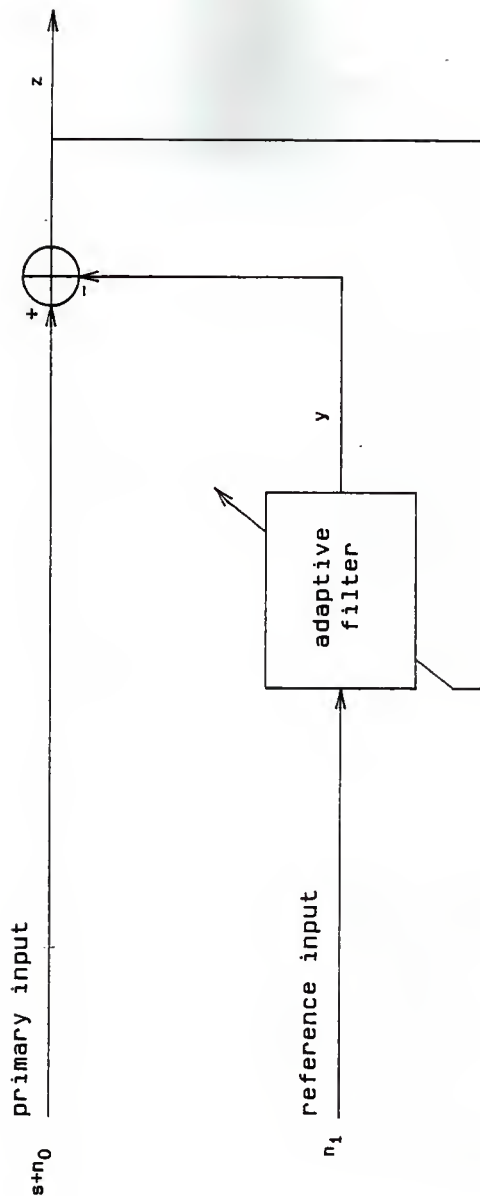


FIGURE 5: Adaptive noise canceller (ANC) .

signal s but is correlated with n_0 in some (possibly) unknown way. The reference input n_1 is filtered by the adaptive filter to produce y which is as close a replica of n_0 as possible. The output z is the difference of the primary input $s+n_0$ and y .

The adaptive filter of the ANC differs from a conventional filter in that its impulse response is automatically adjusted to produce the 'best result' which in this case is noise cancellation. The adjustment, or adaptation, is accomplished through an algorithm that responds to the ANC output. Thus, the ANC will operate under changing conditions or in situations in which the noise can not be characterized (something that a conventional noise cancelling filter depends on). Widrow [1] provides a simple but convincing argument to support this claim.

The ANC output, $z=s+n_0-y$, is referred to as the error of the system. The object of this system is to produce an output z that is the best fit in the least squares sense to the signal s . Therefore, the object is to minimize the mean squared error. It can be shown that doing so does not reduce the signal power of the output but rather it reduces the average power of the quantity n_0-y . This means that minimizing the error must maximize the adaptive filter's objective of transforming n_1 into y . Furthermore, since the signal power remains constant, minimizing the total output power maximizes the output signal-to-noise power ratio.

These arguments can be extended to cases where the primary input and the reference input do not consist of just signal plus

noise and noise, respectively. As long as the reference input is correlated with part of the primary input and uncorrelated with the rest, the two parts of the primary input can be separated.

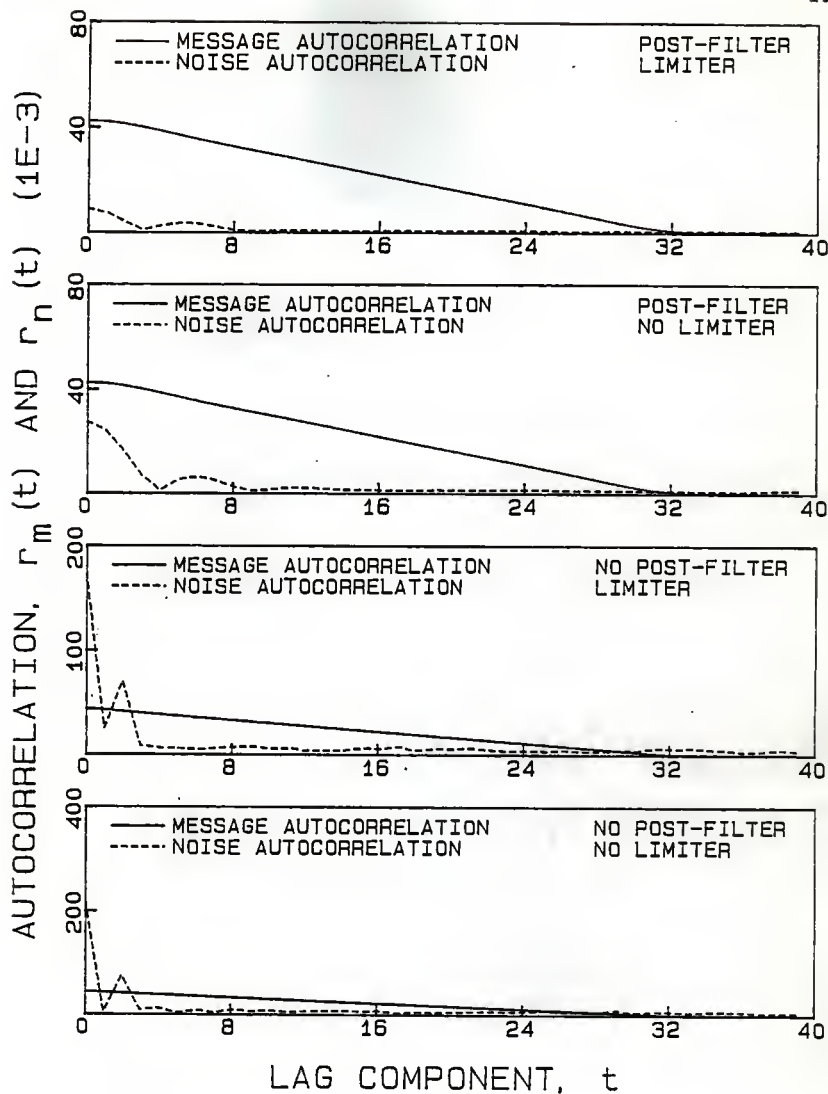
Based on this premise, a scheme was devised to use the ANC to separate the noise from the message in the DLD output. The derivation of the scheme required information about the correlation properties of the noise and message at the output of the DLD. Ratcliffe [2] treats this subject extensively. Another approach at obtaining this information is to use a simulation model as described in the next two sections. This was the method used to obtain the information.

First a noise-free signal consisting of an FSK modulated, pseudo-random sequence was generated. Next this signal was passed through the FSK demodulator. The input to the ANC was then recorded. This input is either the direct output of the DLD or the filtered output of the DLD (depending on whether post-filtering was performed or not). In either case, the input is some variation of the binary message bit sequence. To avoid ambiguities, the result will be referred to as the ANC input. The message autocorrelation estimate was obtained from an ANC input which resulted from a noise-free FSK signal with the estimation being done with FFT techniques. For the noise autocorrelation estimate, some definition of the noise had to be made. The noise at the ANC input was arbitrarily defined as the difference of the input resulting from a noisy FSK signal and the input resulting from a noise-free FSK signal. The final noise

autocorrelation estimate was produced by averaging ten individual estimates. This procedure was completed for several predetection signal-to-noise ratios and for the cases of post-filtering and no post-filtering.

Figure 6 shows the message autocorrelation plotted with the noise autocorrelation for a SNR of 10 dB and with a post-filter bandwidth of 1000 Hz. Figure 6 also shows the same results when the post-filter was omitted. Since the functions are even, only the positive components are shown. Furthermore, the estimates are plotted for lag components 0 to 40 since the rest of the function contains no additional information. For different SNR's, the autocorrelation functions differed only in the scaling of the amplitude of the noise autocorrelation. Note that in both the filtered and non-filtered cases the message is correlated until component number 32, which is the number of samples per bit for this case (the hump past component 32 in the filtered case is caused by the correlation introduced by the post-filter). This result is intuitively obvious since the sequence is pseudo-random. The noise autocorrelation for the filtered case shows the noise is practically uncorrelated past component 20. For the non-filtered case the correlation extends to only the sixth component. Note that the post-filter has eliminated some noise power as evidenced by the difference in peak amplitudes of the noise autocorrelation for the two cases.

Now suppose the primary input of the ANC is the output of the FSK demodulator, i.e., both message and noise. The ANC will



$R = 300$ Hz
 $\Delta f = 300$ Hz
 $SNR = 10$ dB
 $f_s = 9600$ Hz

FIGURE 6: Effect of post-filter and limiter on the correlation properties of the DLD output.

cancel one of these two if the reference input is correlated with one and is uncorrelated with the other. The portion being cancelled will be the one that is correlated with the reference input. Notice that when the DLD output without post-filtering is delayed by 8 samples the noise portion of delayed sequence is uncorrelated with the noise of the non-delayed sequence. Furthermore the message portion of the delayed and non-delayed sequences still remains correlated. Hence the ANC will attempt to cancel the message in the DLD output, i.e. upon adapting, the adaptive filter's output will be the message and the output error will be the noise. The weights of the adaptive filter will have converged once the noise of the error becomes stationary. In this mode, the ANC is being used as a self-tuning filter. This is illustrated in Figure 7.

Suppose the filtered output is considered. The delay would have to be 20 samples instead of 6 for the noise to become uncorrelated. But note that a delay this long weakens the correlation between the signal portion of the two inputs making the job of cancelling more difficult. Thus, omitting the post-filter for this method seems like a good idea especially since determining the correct bandwidth is very difficult as mentioned earlier. One argument for keeping the post filter is that the magnitude of correlation of the noise is reduced, which would improve the results of the cancelling tasks. The initial approach taken is to omit the post-filter when making baud rate estimates with the ANC and to use a delay long enough so the

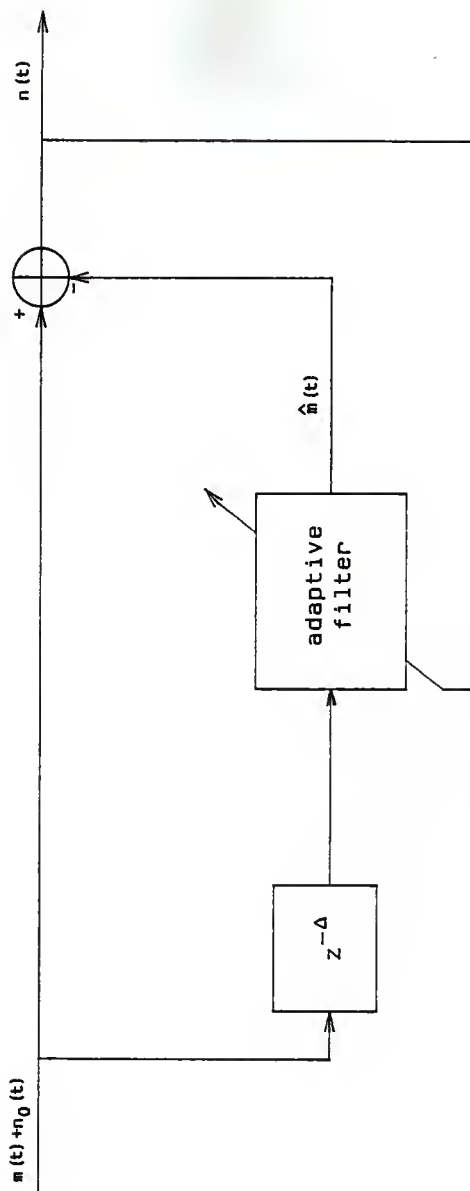


FIGURE 7: The adaptive noise canceller as a self-tuner.

noise is uncorrelated.

Zero-Crossing Estimator

The proposed method of baud rate analysis is to be measured against the traditional method of determining the FSK baud rate, that being the observation of the zero crossings of a demodulated FSK signal. The concept is that a zero crossing in the message should only occur at intervals equal to the baud rate or integral multiples of the baud rate. Obviously noise will perturb the message and thus the zero crossing might not be exactly in the proper place. Then the algorithm should make decisions based on some criteria in order to obtain a baud rate estimate. This algorithm is described in the next section.

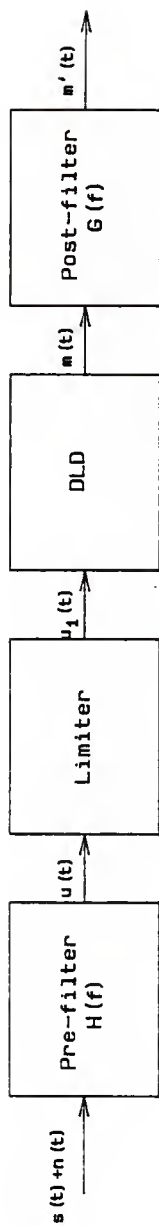
III. Development of a System Model

A mathematical model of the system must be developed in order to simulate the system. Since the simulation is performed on a digital computer, the implications of Shannon's sampling theorem must be kept in mind. Obviously, since the received FSK signal is a bandpass signal, sampling in order to prevent aliasing would require vast amounts of memory for any practical length of received signal. Besides requiring an enormous memory capacity, the simulation would also demand much time to complete. Obviously the modeling cannot be done efficiently using bandpass representations. The solution is to use low-pass modeling techniques. The technique will be applied to the pre-filter, the limiter, and the DLD since they are the only bandpass devices in the system. All other devices are baseband and can be modeled as they are. Thus the FSK demodulator illustrated again in Figure 8a is modeled by its low-pass equivalent representation shown in Figure 8b.

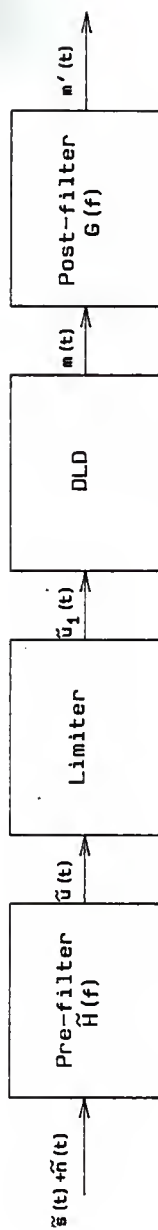
Low Pass Techniques

Low-pass modeling [3] relies on the property that the pertinent information of a band-pass signal is contained in its magnitude and in its phase. Both of these are slowly varying in relation to the carrier frequency and thus are easier to simulate with a digital computer. The relation between a bandpass signal $s(t)$ and its equivalent low-pass counterpart or complex envelope $\tilde{s}(t)$ is, by definition,

$$s(t) = \operatorname{Re}\{\tilde{s}(t)e^{j2\pi f_c t}\} \quad (3.1)$$



a. Bandpass system.



b. Low-pass equivalent representation.

FIGURE 8: FSK demodulator.

If the complex envelope is expressed in terms of its magnitude and argument then it follows that

$$s(t) = \tilde{s}(t) \cos[2\pi f_c t + \phi(t)] \quad (3.2)$$

where $\tilde{s}(t)$ is the magnitude of $s(t)$ and $\phi(t)$ is the argument. Evidently $\tilde{s}(t)$ is a complex function with the property that its magnitude is the envelope of $s(t)$ and its argument is the phase of $s(t)$. All bandpass functions, both deterministic and random, can be put in this format.

Bandpass Pre-Filter

A bandpass filter with an impulse response $h(t)$ has an equivalent low-pass impulse response $\tilde{h}(t)$ which by definition satisfies

$$h(t) = 2 \operatorname{Re}[\tilde{h}(t)e^{j2\pi f_c t}] \quad (3.3)$$

where f_c is the center frequency of the filter. An equivalent low-pass transfer function, $H(f)$, can be derived from this definition [3]. It is

$$H(f) = \tilde{H}(f-f_c) + \tilde{H}^*(-f-f_c) \quad (3.4)$$

where $H(f)$ and $\tilde{H}(f)$ are the Fourier transforms of $h(t)$ and $\tilde{h}(t)$, respectively. Also, $*$ represents conjugation. This relationship can be shifted in frequency to obtain the equivalent low-pass transfer function in terms of the actual bandpass transfer function

$$\tilde{H}(f) = [H(f+f_c)]_{\text{low-pass term}} \quad (3.5)$$

The low-pass model of the filter will have a bandwidth equal to one-half the RF bandwidth, i.e. $B_{lp} = B_{RF}/2$. All transform relations for linear systems are also valid for low-pass models,

e.g., for an input of $X(f)$ and a transfer function $H(f)$ the output $Y(f)$ is

$$\tilde{Y}(f) = \tilde{H}(f)\tilde{X}(f) \quad (3.6)$$

Limiter

The limiter model with input $\tilde{u}(t)$ has an output $\tilde{u}_1(t)$ which is related to the input by

$$\tilde{u}_1(t) = A e^{j \arg[\tilde{u}(t)]} \quad (3.7)$$

where A is some constant amplitude.

Delay-Line Discriminator

The block diagram of the low-pass equivalent model of a delay-line discriminator is shown in Figure 9. The output of the delay block can be characterized in terms of its input by

$$\tilde{u}_2(t) = \tilde{u}_1(t-T) e^{-j2\pi f_c T} \quad (3.8)$$

This can be verified by expanding the definition of the complex envelope of $u_2(t) = u_1(t-T)$.

Each output of the 90° hybrid is equal to the corresponding input minus a 90° phase shifted copy of the remaining input. In equations this can be stated as

$$\tilde{z}_1(t) = \tilde{u}_1(t) - j \tilde{u}_2(t) \quad (3.9a)$$

$$\tilde{z}_2(t) = \tilde{u}_2(t) - j \tilde{u}_1(t) \quad (3.9b)$$

The action of the square law detector is to produce an output proportional to the magnitude squared of the complex envelope of the input, i.e.

$$v_1(t) = |\tilde{z}_1(t)|^2 \quad (3.10a)$$

$$v_2(t) = |\tilde{z}_2(t)|^2 \quad (3.10b)$$

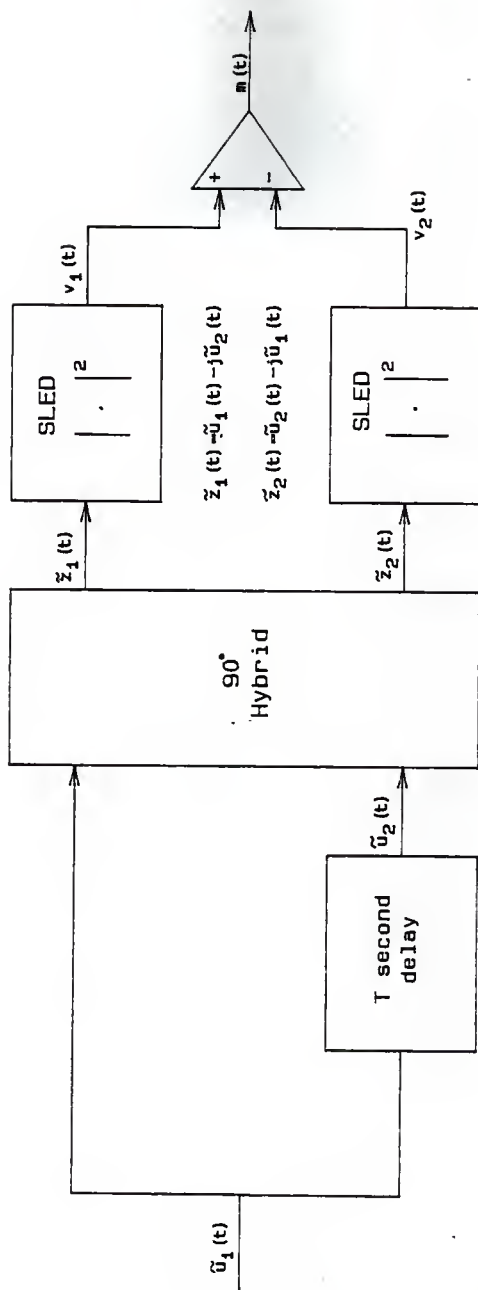


FIGURE 9: Low-pass equivalent model of a delay-line discriminator.

Note that the output is no longer a bandpass signal. Thus the low-pass model is the same as the physical system from this point on.

Finally the output of the DLD is simply the difference between the two square law detectors

$$m(t) = v_1(t) - v_2(t) \quad (3.11)$$

The discriminator equation is derived as follows. The outputs of the 90° hybrid in terms of the input $\tilde{u}_1(t)$ are

$$\tilde{z}_1(t) = \tilde{u}_1(t) - j \tilde{u}_1(t-T) e^{-j2\pi f_c T} \quad (3.12a)$$

$$\tilde{z}_2(t) = \tilde{u}_1(t-T) e^{-j2\pi f_c T} - j \tilde{u}_1(t) \quad (3.12b)$$

From these, the output of the first square law envelope detector is determined to be

$$v_1(t) = |\tilde{z}_1(t)|^2 = \tilde{z}_1(t) \tilde{z}_1^*(t) \quad (3.13a)$$

$$= [\tilde{u}_1(t) - j \tilde{u}_1(t-T) e^{-j2\pi f_c T}] [\tilde{u}_1^*(t) + j \tilde{u}_1^*(t-T) e^{j2\pi f_c T}] \quad (3.13b)$$

$$= |\tilde{u}_1(t)|^2 + j \tilde{u}_1(t) \tilde{u}_1^*(t-T) e^{j2\pi f_c T} + |\tilde{u}_1(t-T)|^2 - j \tilde{u}_1^*(t) \tilde{u}_1(t-T) e^{-j2\pi f_c T} \quad (3.13c)$$

Similarly

$$v_2(t) = |\tilde{z}_2(t)|^2 = j \tilde{u}_1(t) \tilde{u}_1^*(t-T) e^{j2\pi f_c T} + |\tilde{u}_1(t-T)|^2 + j \tilde{u}_1^*(t) \tilde{u}_1(t-T) e^{-j2\pi f_c T} \quad (3.14)$$

The comparator output is then

$$m(t) = v_1(t) - v_2(t) \quad (3.15a)$$

$$= 2j \tilde{u}_1(t) \tilde{u}_1^*(t-T) e^{j2\pi f_c T} - 2j \tilde{u}_1^*(t) \tilde{u}_1(t-T) e^{-j2\pi f_c T} \quad (3.15b)$$

$$= 2[\tilde{u}_1(t)\tilde{u}_1^*(t-T)e^{j2\pi f_c T}e^{j\frac{\pi}{2}} + \tilde{u}_1(t)\tilde{u}_1^*(t-T)e^{j2\pi f_c T}e^{j\frac{\pi}{2}}] \quad (3.15c)$$

$$= 2(\tilde{u}_1(t)\tilde{u}_1^*(t-T)e^{j2\pi f_c T}e^{j\frac{\pi}{2}} + [\tilde{u}_1(t)\tilde{u}_1^*(t-T)e^{j2\pi f_c T}e^{j\frac{\pi}{2}}]^*) \quad (3.15d)$$

$$= 4\text{Re}(\tilde{u}_1(t)\tilde{u}_1^*(t-T)e^{j(2\pi f_c T + \frac{\pi}{2})}) \quad (3.15e)$$

To observe the discriminator effect for a continuous-wave signal, let the input be

$$u_1(t) = A \cos[2\pi(f_c + \Delta f)t] \quad (3.16)$$

then

$$\tilde{u}_1(t) = A e^{j2\pi\Delta f t} \quad (3.17)$$

Substituting into the output expression for a general input, (3.15e), gives

$$m(t) = 4\text{Re}([Ae^{j2\pi\Delta f t}][Ae^{-j2\pi\Delta f(t-T)}e^{j(2\pi f_c T + \frac{\pi}{2})}) \quad (3.18)$$

Simplifying yields

$$m(t) = 4A^2 \cos[2\pi(\Delta f + f_c)T + \pi/2] \quad (3.19a)$$

$$= -4A^2 \sin[2\pi(\Delta f + f_c)T] \quad (3.19b)$$

This is the result given and discussed in the DLD description of the previous section.

It is important to note that the input $\tilde{u}_1(t)$ will consist of two terms in the general case, a signal envelope and a noise envelope. Ratcliffe [2] discusses the statistics and the power spectra associated with the general case.

Low-pass Post-Filter

The post-filter, being low-pass, can be implemented directly into the system model. No equivalent model is necessary.

Adaptive Noise Canceller

The adaptive filter of the canceller is so named because the filter weights adapt to changing conditions. The implementation of the adaptive filter and the algorithm for the weight adjustment will now be discussed. The discussion is an abbreviated version of the one in Widrow [1].

The adaptive filter is modeled with the adaptive linear combiner shown in Figure 10. Note that X_j and W are vectors, while y_j and d_j are scalars. The vectors are defined as

$$X_j = [x_{0j} \ x_{1j} \ \dots \ x_{nj}]^T \quad (3.20)$$

$$W = [w_0 \ w_1 \ \dots \ w_n]^T \quad (3.21)$$

Generally x_{0j} is set to 1 so that w_0 can be considered the biasing weight. The output y_j is the inner product of X_j and W . The error e_j is the difference between the desired response d_j and the actual response y_j . That is

$$y_j = X_j^T W = W^T X_j \quad (3.22)$$

$$e_j = d_j - y_j = d_j - W^T X_j \quad (3.23)$$

Note that in accordance with the system description of the ANC, $d_j = x_{j+\Delta}$ where Δ is the delay in number of samples between the primary input and the reference input.

Since the desire is to minimize the mean squared error, the mean squared error is derived first. It is

$$E[e_j^2] = E[d_j^2] - 2E[d_j X_j^T] W + W^T E[X_j X_j^T] W \quad (3.24)$$

where $E[\cdot]$ denotes expectation. The gradient of the error function, denoted by V , is obtained by differentiating this result with respect to W . The gradient is

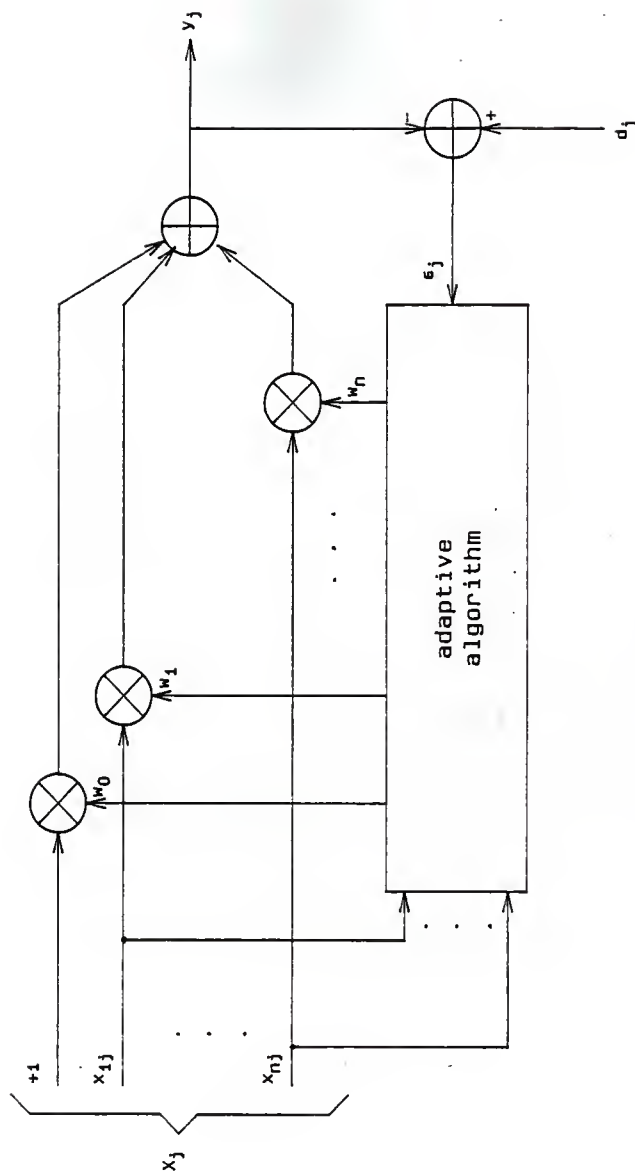


FIGURE 10: Adaptive linear combiner.

$$\mathbf{V} = \mathbf{V}[E[e_j^2]] = -2E[d_j \mathbf{x}_j^T] + 2E[\mathbf{x}_j \mathbf{x}_j^T] \mathbf{W} \quad (3.25)$$

The optimal weight vector $\mathbf{W}^*$ is obtained by setting the gradient to zero.

$$\mathbf{W}^* = \{E[\mathbf{x}_j \mathbf{x}_j^T]\}^{-1} E[d_j \mathbf{x}_j^T] \quad (3.26)$$

The idea behind using the gradient method is that the mean squared function is a quadratic function of the weights. It can be pictured as a concave hyperparaboloidal surface that never goes negative, i.e. a bowl resting on some point in the hyper-space. The object is to get to the bottom of the bowl, at which point the gradient is the zero vector.

The LMS algorithm implements the method of steepest descent. According to this method, the next weight vector estimate $\mathbf{W}_{j+1}$ is equal to the present weight vector estimate $\mathbf{W}_j$ plus a change proportional to the negative of the gradient; i.e.,

$$\mathbf{W}_{j+1} = \mathbf{W}_j - \mu \mathbf{V}_j \quad (3.27)$$

The parameter μ controls stability and convergence rate. The gradient $\mathbf{V}_j$ is the true gradient evaluated with the weight vector of the j^{th} iteration.

The LMS algorithm assumes e_j^2 is an estimate of the mean squared error $E[e_j^2]$. The gradient is then taken of the estimate. This gradient, denoted by $\hat{\mathbf{V}}_j$, is an estimate of the true gradient and can be determined to be

$$\hat{\mathbf{V}}_j = \mathbf{V}[e_j^2]_{\mathbf{W}=\mathbf{W}_j} = -2e_j \mathbf{x}_j \quad (3.28)$$

Using this estimate in place of the true gradient in (3.27) gives

$$\mathbf{W}_{j+1} = \mathbf{W}_j + 2\mu e_j \mathbf{x}_j \quad (3.29)$$

The adaptive filter of Figure 7 is implemented with a tapped

delay line as shown in Figure 11. The input signal vector is

$$\mathbf{X}_j = [x_j \ x_{j-1} \ \dots \ x_{j-n+1}]^T \quad (3.30)$$

The components of this vector are the delayed components of $x_{j+\Delta}$ which is a sample of the time series $\dots, x_{j+\Delta-1}, x_{j+\Delta}, x_{j+\Delta+1}, \dots$.

When implementing this algorithm the parameter μ must be specified. If made too large, the algorithm becomes unstable. If μ is made too small, the weight vector takes too much time to converge to the optimal weight vector. Although there are ways to determine the proper μ to insure stability, it is generally easier to use a trial-and-error procedure in this problem. This was the manner used to set μ for fast convergence in the simulation.

Besides convergence rate and stability, another concern in picking μ is imbedded in the LMS equation, Eqn. (3.27). There are two sources of information that contribute to update of the weight vector. One is the current weight vector which represents a history of the weight vectors due to the regressive structure of Eqn. (3.27). The second is the gradient which is estimated by the the product of the current filter input vector and the error. The choice of μ decides what emphasis should be given to each. If μ is made small then the emphasis is placed on the current weight vector. In this instance a weight vector with many weights is useful since it can retain more information. If μ is made large then the emphasis is placed on the gradient and the information contained in previous weight vectors is soon lost.

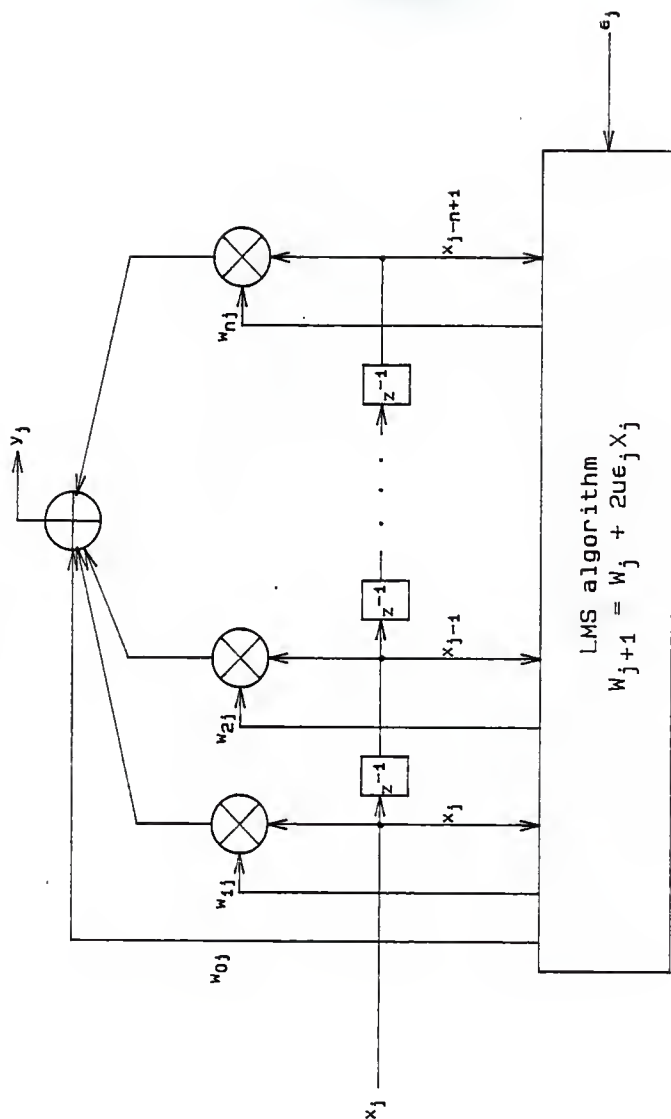


FIGURE 11: The LMS adaptive filter implemented with a tapped delay line.

In this case a large number of weights would not be beneficial.

For the band rate analysis system, both reasons of choice have merit. Since the message sequence is pseudo-random, the mean of the ANC input will also be pseudo-random. Making μ large corresponds to attempting to obtain good noise cancellation. The reason μ must be large is that convergence should occur before the mean value changes. The output would then be the clean message sequence which would easily indicate the band rate. Making μ small will not provide good noise cancellation, but rather, the weight vector will contain the information about the sequence. Suppose the mean and variance of the changing weight vectors is determined. An intuitive argument is that the mean should resemble the autocorrelation function of the message displaced by the delay of the ANC. Once again the band rate could be easily determined. As for the variance, a relative maximum should occur at the first sample of the second bit interval (again a displacement results from the delay) resulting from the indication of whether the current bit has been repeated or not.

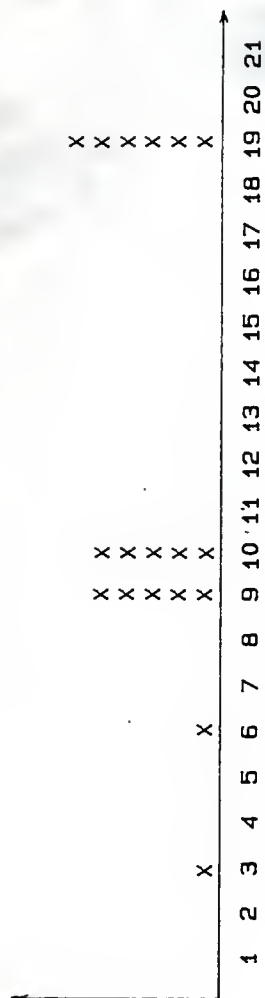
Zero-Crossing Algorithm

The zero-crossing algorithm is implemented as follows. First a set is constructed whose members are the number of samples between each zero crossing. Next a histogram is formed from the data of the set. Since it is very possible that the bit time will not relate to an integral number of samples, bins of the histogram around the nearest integral sample which represents the bit time might also have entries. For example, if the band

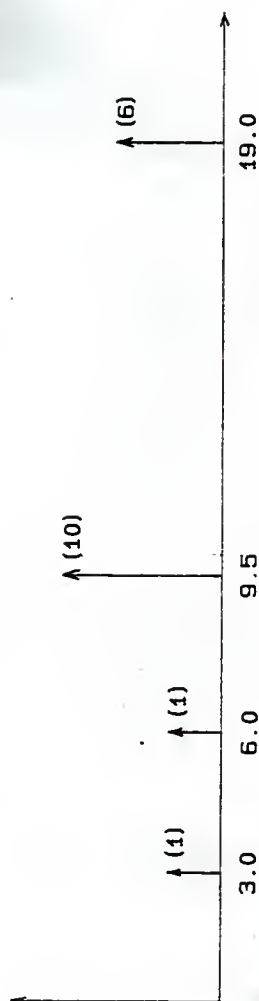
rate is 2 bps and the sampling rate is 19 Hz the resulting histogram might look like the one illustrated in Figure 12a. Noise perturbations may also cause this result, especially for low signal-to-noise ratios. Noise may also introduce false crossings as illustrated in bins number 3 and 6.

To negate this spilling action, the entries of each group are averaged. A group is defined as a collection of bins not containing two or more empty, adjacent bins. These averages are then plotted on the result line as impulses with weights equal to the number of entries that contributed to the respective average. When applied to the histogram of Figure 12a, this procedure results in the plot of Figure 12b. Under optimum conditions, the result will be a group of impulses with the distance between adjacent impulses being the bit time normalized by the sampling interval (or conversely, the sampling rate divided by the bit rate). The leftmost impulse will be located at the normalized bit time and the others at integral multiples of the bit rate. When conditions are not so good, as in Figure 12b, some sort of decisions must be made.

There are several criteria which the decision must satisfy. Before listing the criteria, the definition of consistent and inconsistent result should be stated. A consistent result occurs when the location of an impulse to the right of the considered impulse is an integral multiple of the location of the considered impulse, within some tolerance. An inconsistent result occurs when the tolerance is exceeded. The number of



a. Probable histogram



b. Resulting impulse distribution.

FIGURE 12: Histogram analysis of the zero-crossing estimator.

consistencies is the sum of the weights of consistent impulses while the sum of all weight of inconsistent impulses is the number of inconsistencies.

The first criterion is that the weight of the considered impulse plus the number of consistencies must be as large as some number times the number of inconsistencies. The second criterion is that the weight of the considered impulse must be at least a certain proportion of the sum of all impulse weights (which is the total number of zero crossings). The final criterion is that the number of consistencies must be at least a certain proportion of the sum of weights (number of zero crossings). Obviously there is a tradeoff when setting the tolerance and the ratios. The optimal parameters will be determined during initial runs of the simulation. If no impulse meets the criteria, the impulse used for the bit time estimate is the one of maximum weight.

Consider the example in Figure 12 once again. The first impulse meeting the requirements (upon reasonable choice of the parameters) is the one at 9.5. The bit time estimate is then found by multiplying the location by the sampling interval. For the example the bit time estimate would be 0.5 seconds since the sampling interval is 1/19 seconds. Finally the band rate estimate is found as the inverse of the bit time which would be 2 bps for the example.

IV. Simulation Description

The system modeled in the previous section will be subjected to a Monte Carlo simulation to determine its performance. The first step of the simulation is to generate an FSK signal. Next noise must be added subject to the signal-to-noise ratio parameter. Then the response of the system to the signal-plus-noise input is observed. This procedure is repeated many times so that the error between the system response and the known signal parameters can be averaged in a sound statistical sense. The results will be the mean of the band rate estimate and the variance of the estimate.

FSK Signal Modulator

In order to simulate the band rate analysis system, a representation for a received FSK signal must be formulated. The scheme used for the simulation is based on work done by Hammels and Rstcliffe [4]. The input to the modulator is a binary message sequence having symbols that are equally likely and statistically independent. To meet these conditions a pseudorandom bit sequence is used with a bit time of T_b seconds. Each T_b seconds the modulator puts out one of two different waveforms in accordance to the FSK scheme. The two waveforms, which are of duration T_b seconds, are

$$e_{1,2}(t) = A \cos[2\pi(f_c \pm f_{FSK})t] \quad (4.1)$$

The plus sign is used for a 1 in the bit sequence and the minus is used for a 0. The frequency deviation from the center frequency is denoted by f_{FSK} . The FSK signal is then a composite

of the separate waveforms. It can be expressed as

$$s(t) = \sum_{n=-\infty}^{\infty} s_n(t-nT_b) \quad (4.2)$$

where the n^{th} transmission $s_n(t-T_b)$ is one of the two waveforms of Eqn. (4.1).

For this FSK representation, the signal will have a phase discontinuity at the end of each bit interval unless the two signaling frequencies, $f_c + f_{\text{FSK}}$ and $f_c - f_{\text{FSK}}$, contain an integral number of whole cycles per bit interval. This discontinuity will occur even when one bit is repeated. Generally FSK is generated using two oscillators, one operating at each of the different signalling frequencies. The output is produced with a switch that connects to either one oscillator or the other. The switch is controlled by the bit sequence. In this manner, the phase of each oscillator output will be continuous when a bit is repeated. This form was used for the simulation since it is a commonly used scheme. The equation used to implement this FSK scheme is

$$a(t) = \sum_{n=-\infty}^{\infty} s_n(t) \{u(t-nT_b)[1-u(t-nT_b-T_b)]\} \quad (4.3)$$

This time $s_n(t)$ is not limited to a duration of T_b seconds, instead its duration will be for the entire signalling interval. The unit step functions, represented by $u(t)$, create a window from time nT_b to $(n+1)T_b$ and thus implement the switching.

To be useful the resulting bandpass FSK signal, $a(t)$, should be expressed in its low-pass representation, $\tilde{a}(t)$. To begin this development each waveform $s_n(t)$ has a complex envelope that is

essily shown to be

$$\tilde{s}_{1,2}(t) = Ae^{\pm j2\pi\Delta f_{FSK}t} \quad (4.4)$$

Then by definition and substitution each waveform can be expressed in terms of its complex envelope as

$$s_n(t) = \text{Re}\{\tilde{s}_n(t)e^{j2\pi f_c t}\} \quad (4.5a)$$

$$= \text{Re}\{Ae^{\pm j2\pi\Delta f_{FSK}t} e^{j2\pi f_c t}\} \quad (4.5b)$$

Substituting the definition result into the FSK representation gives

$$s(t) = \sum_{n=-\infty}^{\infty} \text{Re}\{\tilde{s}_n(t)e^{j2\pi f_c t}\} n(t-nT_b)[1-n(t-nT_b-T_b)] \quad (4.7)$$

Moving the summation inside the equation yields

$$s(t) = \text{Re}\{e^{j2\pi f_c t} \sum_{n=-\infty}^{\infty} \tilde{s}_n(t)n(t-nT_b)[1-n(t-nT_b-T_b)]\} \quad (4.8)$$

Now the complex envelope of $s(t)$ is seen to be

$$\tilde{s}(t) = \sum_{n=-\infty}^{\infty} \tilde{s}_n(t)n(t-nT_b)[1-n(t-nT_b-T_b)] \quad (4.9)$$

Then substituting Eqn. (4.4) into this result yields a low-pass form of an FSK signal that can easily be implemented in the simulation. The equation for implementation is

$$\tilde{s}(t) = \sum_{n=-\infty}^{\infty} Ae^{\pm j2\pi\Delta f_{FSK}t} n(t-nT_b)[1-n(t-nT_b-T_b)] \quad (4.10)$$

Obviously the limits of the summation become finite for a finite length message. Once again the $\pm$ distinguishes between either a 0 or a 1 being sent.

Noise Model

The FSK signal generated for simulation purposes by the

expression above will have noise added to it before entering the system. Since the object of the investigation is to determine how well one method of band rate analysis performs at low signal-to-noise ratios (SNR), the manner of setting the SNR will be discussed. The formal definition of the SNR is the ratio of signal power to noise power as measured after the pre-filter. Note that SNR is not meaningful until the pre-filter bandwidth is specified. The noise is assumed to be from a white Gaussian random process.

Suppose a random sample, $x(m)$, of length N is drawn from a statistically independent, Gaussian random process with zero mean. The average power in the sample, P_x , is σ^2 , the variance of each sample. This can be expressed as

$$P_x = \sigma^2 = \frac{1}{N} \sum_{m=0}^{N-1} \overline{x^2(m)} \quad (4.11)$$

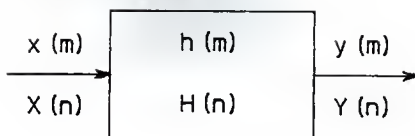
where the overbar denotes mean value. Parseval's theorem for the discrete Fourier transform [5] which is defined by

$$X(n) = \sum_{m=0}^{N-1} x(m) e^{-j2\pi mn/N} \quad (4.12)$$

states that (4.11) can also be expressed as

$$\sigma^2 = \frac{1}{N^2} \sum_{n=0}^{N-1} \overline{|X(n)|^2} \quad (4.13)$$

It can be shown that the mean value of $|X(n)|^2$ is independent of n and thus is $N\sigma^2$. Suppose the noise sample is passed through a low-pass filter as illustrated below.



The power response of the filter is

$$|H_n|^2 = \begin{cases} |H(nf_0)|^2 & k=0,1,\dots,N/2 \\ |H((n-N)f_0)|^2 & k=N/2+1,\dots,N-1 \end{cases} \quad (4.14)$$

where f_0 is the fundamental frequency of the time sample. Then the output power, P_y , is

$$P_y = \frac{\sigma^2}{N} \sum_{n=0}^{N-1} |H_n|^2 \quad (4.15)$$

To simplify this expression, the equivalent noise bandwidth for the filter can be defined as

$$2B_n = \frac{1}{H_0} \sum_{n=0}^{N-1} |H_n|^2 f_0 \quad (4.16a)$$

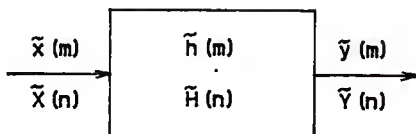
$$= \frac{1}{T_s N H_0} \sum_{k=0}^{N-1} |H_n|^2 \quad (4.16b)$$

This is actually an approximation which grows better as N grows larger. Note that $f_0 = 1/(T_s N)$ where T_s is the sampling interval. Substituting this result into (4.15) and solving for σ^2 gives

$$\sigma^2 = \frac{P_y}{2H_0^2 T_s B_n} \quad (4.17)$$

These results are for the low-pass case. Since noise envelopes used for the bandpass case are complex and have, in gene-

ral, both real and imaginary parts, it will be necessary to extend these concepts to the low-pass equivalent model as shown below.



For the low-pass model, the powers $P_{\tilde{y}}$ and P_y are related by

$$P_y = \frac{1}{2} P_{\tilde{y}} \quad (4.18)$$

This can be shown by taking the mean squared value of the expression relating $y(t)$ and $\tilde{y}(t)$.

Now each sample of the complex noise envelope $\tilde{x}(m)$ requires two numbers from the random process, one for the real part and one for the imaginary part. The object is to find a relation between σ_x^2 and σ_i^2 to the output power P_y of the bandpass process $\tilde{y}(t)$. The case of interest is where $\tilde{x}_r(t)$ and $\tilde{x}_i(t)$ are independent processes with identical statistical properties so that $\sigma_x^2 = \sigma_i^2 = \sigma^2$. Since

$$\tilde{y}(t) = \tilde{x}(t) * \tilde{h}(t) \quad (4.19a)$$

$$= [\tilde{x}_r(t) + j\tilde{x}_i(t)] * [\tilde{h}_r(t) + j\tilde{h}_i(t)] \quad (4.19b)$$

(* denotes convolution) it is obvious that

$$\tilde{y}_r(t) = \tilde{x}_r(t) * \tilde{h}_r(t) - \tilde{x}_i(t) * \tilde{h}_i(t) \quad (4.20a)$$

and

$$\tilde{y}_i(t) = \tilde{x}_i(t) * \tilde{h}_r(t) + \tilde{x}_r(t) * \tilde{h}_i(t) \quad (4.20b)$$

In many cases $\tilde{h}(t)$ will be a real function so that

$$\tilde{y}_r(t) = \tilde{x}_r(t) * \tilde{h}_r(t) \quad (4.21a)$$

$$\tilde{y}_i(t) = \tilde{x}_i(t) * \tilde{h}_i(t) \quad (4.21b)$$

Recall that the low-pass procedure derived earlier can be applied to these expressions. Hence

$$\sigma_{\tilde{x}_r}^2 = P_{\tilde{y}_r} \frac{1}{2T_s B_n H_0^2} \quad (4.22a)$$

$$\sigma_{\tilde{x}_i}^2 = P_{\tilde{y}_i} \frac{1}{2T_s B_n H_0^2} \quad (4.22b)$$

Note that B_n is the equivalent low-pass bandwidth which is one-half the bandpass bandwidth, i.e. $B_n = B_{RF}/2$.

Then since the variances have been assumed equal, it follows from

$$P_{\tilde{y}} = P_{\tilde{y}_r} + P_{\tilde{y}_i} = 2P_y \quad (4.23)$$

that

$$P_{\tilde{y}_r} = P_{\tilde{y}_i} = P_y \quad (4.24)$$

The conclusion is to set $\sigma_{\tilde{x}_r}^2 = \sigma_{\tilde{x}_i}^2 = \sigma^2$ where

$$\sigma^2 = \frac{P_y}{2T_s B_n H_0^2} \quad (4.25)$$

The final step is to relate the noise power P_y , the noise variance σ^2 , and the SNR. Since the FSK signal is a sinusoid of amplitude A at one of two frequencies, the signal power is $A^2/2$. Therefore

$$\text{SNR} = \frac{A^2/2}{P_y} \quad (4.26)$$

Or

$$P_y = \frac{A^2/2}{\text{SNR}} \quad (4.27)$$

The final result is obtained by substituting this expression for P_y into (4.24). Thus, the noise should have zero mean and variance defined by

$$\sigma^2 = \frac{A^2/2}{T_s B_n (\text{SNR})} \quad (4.28)$$

Or since $B_n = B_{\text{RF}}/2$,

$$\sigma^2 = \frac{A^2}{T_s B_{\text{RF}} (\text{SNR})} \quad (4.29)$$

V. Preliminary Observations and Estimator Definitions

The object of the investigation is to determine the band rate of an FSK signal using the adaptive noise canceller (ANC). In order to do so, first a decision has to be made as to how the band rate can be obtained from the information provided by the ANC. Since the ANC is a non-linear device (linearity only occurs once convergence of the weight vector is reached and maintained), this decision should be made after the outputs of the ANC for different parameters and situations are observed. The observations will be presented in this section along with the decision of how to obtain the band rate from the ANC information.

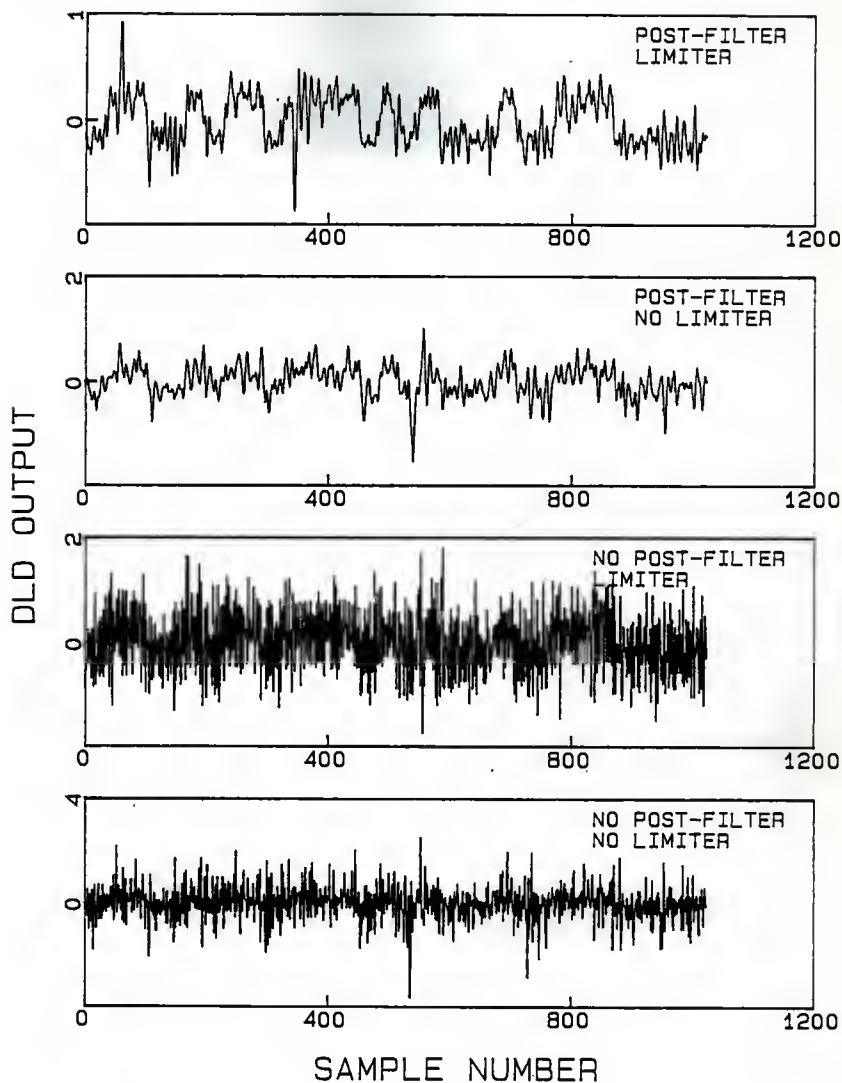
To generate the signal, the band rate for the FSK generation is arbitrarily set to 300 Hz. The frequency deviation will be initially set to 300 Hz. Generally for FSK systems the frequency deviation must be as large as the band rate to insure quality reception. As deviation increases, performance degrades also. Therefore this represents the worst case situation for a given SNR. The signal is sampled at a rate of 9600 Hz and 1024 samples are recorded. This means that 32 bits will be seen in the sampled sequence and each bit will consist of 32 samples. As discussed earlier, the system is designed to operate for FSK signals with signal bandwidth of 6000 Hz or less and band rates of 1000 Hz or less. Consequently the pre-filter bandwidth is set to 6000 Hz and the post-filter bandwidth is set to 1000 Hz. In order that the DLD operates in the linear region for this bandwidth, the parameters of the DLD are $T = 13.854 \text{ msec}$ and $n =$

266 (see Egnas. (2.3) and (2.6)). As for the limiter and post-filter, they will be included in the demodulator system. The reason for doing so will be substantiated shortly. These parameters are held constant throughout the observations unless otherwise noted. Furthermore, unless otherwise specified, the signal-to-noise ratio is set to 8 dB.

DLD Output

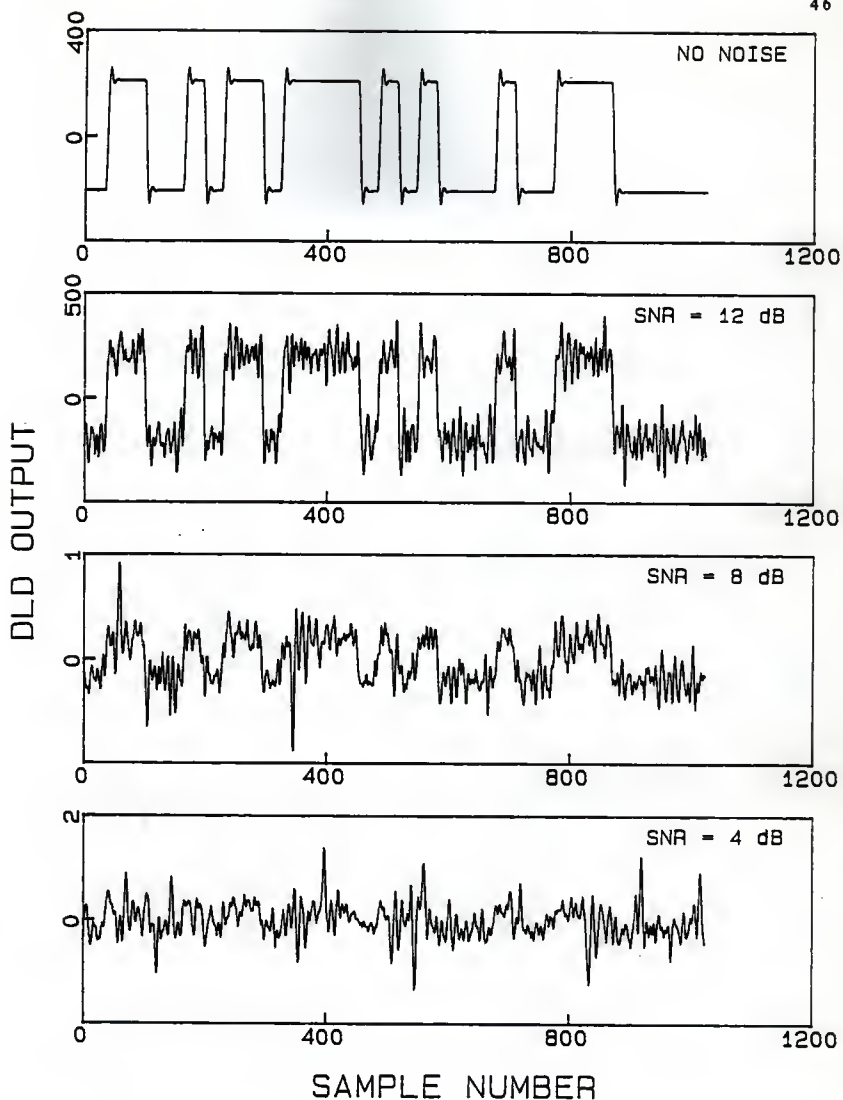
It was previously argued that the post-filter should be omitted when the ANC approach for band rate determination is used (see the ANC discussion in section II). Figure 13 suggests that the post-filter should be included. The four plots in Figure 13 are the DLD output for the aforementioned parameter set except as noted on each plot. The noise sequence added to the FSK signal is the same for each case. These plots investigate the effect of the post-filter and the limiter upon the output. Even for modest bandwidth of 1000 Hz, the post-filter clearly improves the output. As for the limiter, when the post-filter is included, it too is desirable. When the post-filter is omitted the use of the limiter is questionable. Based on these observations, the post-filter and limiter should be included in the system.

The output of the DLD (post-filter and limiter included) for four cases is plotted in Figure 14. The four cases represent a noise-free FSK signal, and signals with SNR of 12 dB, 8 dB and 4 dB. As should be expected, the signal is degraded as the signal-to-noise ratio decreases. For SNR = 12 dB the message sequence can be easily identified. When the SNR drops to 8 dB the dis-



$R = 300 \text{ Hz}$
 $\Delta f = 300 \text{ Hz}$
 $\text{SNR} = 8 \text{ dB}$
 $f_s = 9600 \text{ Hz}$

FIGURE 13: Effect of post-filter and limiter on DLD output.



$R = 300$ Hz
 $\Delta f = 300$ Hz
 $f_s = 9600$ Hz

FIGURE 14: DLD output for signals of small deviation and varying SNR.

inction between the ones and zeroes is less obvious. At SNR = 4 dB, the bits are indistinguishable.

Figure 15 shows the DLD output for the same signal-to-noise ratios but with the frequency deviation set to 2000 Hz. This is more likely the case for practical systems than the minimum deviation as shown in Figure 14. The ones and zeroes are now clearly discernible at SNR = 8 dB but only vaguely discernible when the SNR is 4 dB.

ANC Investigation

The input of the ANC, implemented as in Figure 7, consists of the output sequence from the DLD. The output of the ANC can be thought of as consisting of three sequences, the error sequence, the adaptive filter output sequence, and the weight vector. The object is to fix the parameters of the ANC such that the baud rate can be estimated from one of these three sequences more accurately than it can be from the input sequence.

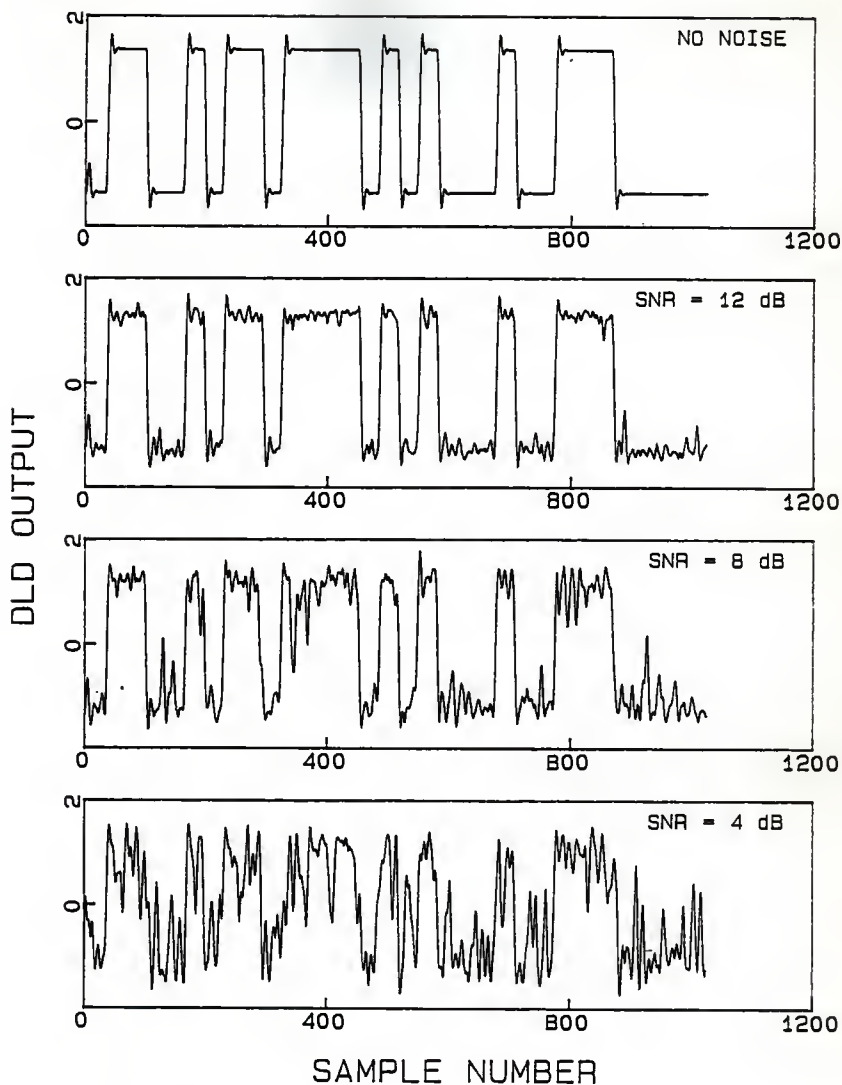
The method postulated in the previous section is that the message and the noise can be separated by the ANC. This relies on the earlier made observation that the noise portion of the input sequence when delayed 8 or more samples (with the post-filter and limiter) is uncorrelated with the noise of the non-delayed input but that the message portion is still correlated.

Recall the LMS equation used to implement the method. It is

$$\mathbf{w}_{j+1} = \mathbf{w}_j - \mu \hat{\mathbf{v}}_j \quad (5.1)$$

where the estimate for the gradient is

$$\hat{\mathbf{v}}_j = -2e_j \mathbf{x}_j \quad (5.2)$$



$R = 300 \text{ Hz}$
 $\Delta f = 2000 \text{ Hz}$
 $f_s = 9600 \text{ Hz}$

FIGURE 15: DLD output for signals of large deviation and varying SNR.

In order for the cancelling (actually predicting in this situation) to perform well the gradient constant μ must be large so that the weight vector converges rapidly. Also, since the message is pseudo-random, the ANC will not be able to predict past one bit. Therefore, convergence needs to occur within one bit time. Thus, it is best to pick μ as large as possible so that stability is maintained. Furthermore, the bias weight should also be included since it allows the weight vector to respond quickly to changes in the input level. There are methods to determine the proper μ [1] but the choice is easier made with trial-and-error procedure. For a weight vector composed of 128 weights μ can be as large as 0.001 without causing instability (maximum μ is dependent of the weight vector length in an inverse relation).

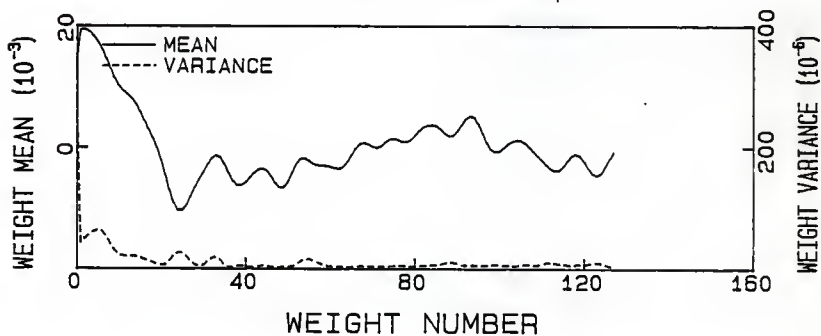
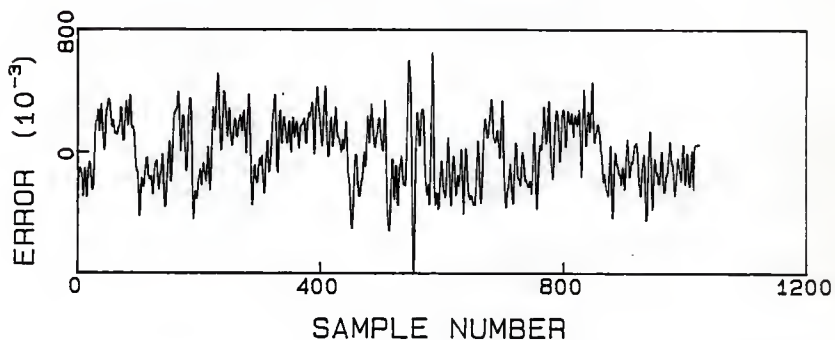
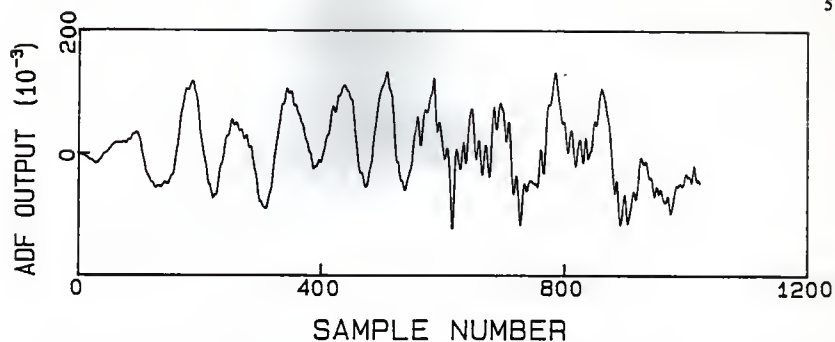
The error sequence can be inspected to see how well the ANC performed the cancelling. Since the message is the portion being cancelled, under optimal cancelling the error would be the noise portion of the input. Therefore, if the error seems to have a stationary mean, the ANC provided good cancellation. This is synonymous to good prediction, i.e. the adaptive filter output should be the message portion of the input.

Recall that for the sampling frequency of 9600 Hz and a baud rate of 300 Hz there will be 32 samples per bit. However the system is designed for baud rates up to 1000 Hz which would have 9.6 samples per bit. Besides the problem of attaining convergence in the time of 9.6 samples another problem exists. Since a

signal with a 1000 Hz band rate will be uncorrelated after the tenth sample for the given sampling frequency, the delay of 8 samples might decorrelate the message so much that cancelling can not be performed. A closer examination of the results should be made in setting this delay. Figures 16a and 16b provide additional insight. Figure 16a consists of the three output sequences of the ANC for the case of $\Delta = 8$. The other parameters are the standard set with $\mu = 0.001$ and $N = 128$. In Figure 16b the delay has been changed to $\Delta = 1$ while all other parameters are the same as in Figure 16a.

The main conclusion to draw from Figures 16a and 16b is that the output sequences for the different choices of delay are very similar. This means that the delay can be set to one sample and the results should not be degraded. By being able to do this, the problems of convergence associated with the larger band rates should not be as severe. One should recall that decreasing the delay increases the correlation of the noise and that this should make matters worse. However the correlation of the message is also increased. Referring back to Figure 6 for the post-filter and limiter case, the slope of the message autocorrelation is larger than the slope of the noise autocorrelation. Thus, the correlation of the message increases faster than that of the noise as the delay is reduced.

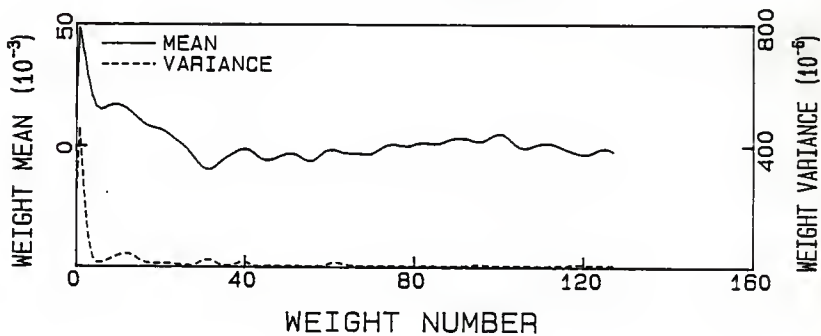
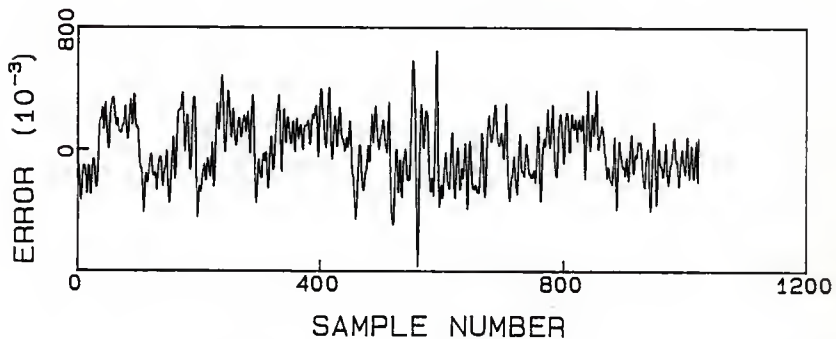
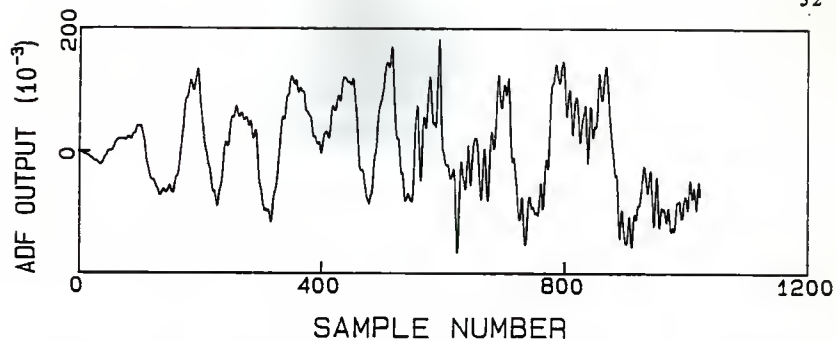
Referring again to Figures 16a and 16b, note how the error sequence dictates the degree of convergence. In both cases the error initially has distinct mean values. After the weights are



R = 300 Hz
 Δf = 300 Hz
 SNR = 8 dB
 f_s = 9600 Hz

DELAY = 8
 MU = $0.1E-2$
 N = 128
 BIAS

FIGURE 16a: ANC output information for varying delay.



$R = 300$ Hz
 $\Delta f = 300$ Hz
 $SNR = 8$ dB
 $f_s = 9600$ Hz

DELAY = 1
 $MU = 0.1E-2$
 $N = 128$
 BIAS

FIGURE 16b: ANC output information for varying delay.

given some time to adjust from their all-zero initial value, the mean of the error does not vary as much. Although the two error sequences are nearly the same, the end of the error sequence for the delay of 8 seems to fluctuate slightly more than when the delay is 1. This suggests convergence is more complete for the delay of 1. However, the ADF output is a bit more ragged for the smaller delay than for the larger delay. Still this is a considerable improvement over the original input which is depicted in the top plot of Figure 13. Another point is that the weight vectors appear to be distinctly different. (Weight vector in this context implies the mean and variance of the individual weight sequences of the iterative update.) A second look might suggest that the weight sequence for the larger delay is just the weight sequence of the smaller delay advanced by the difference in the delay.

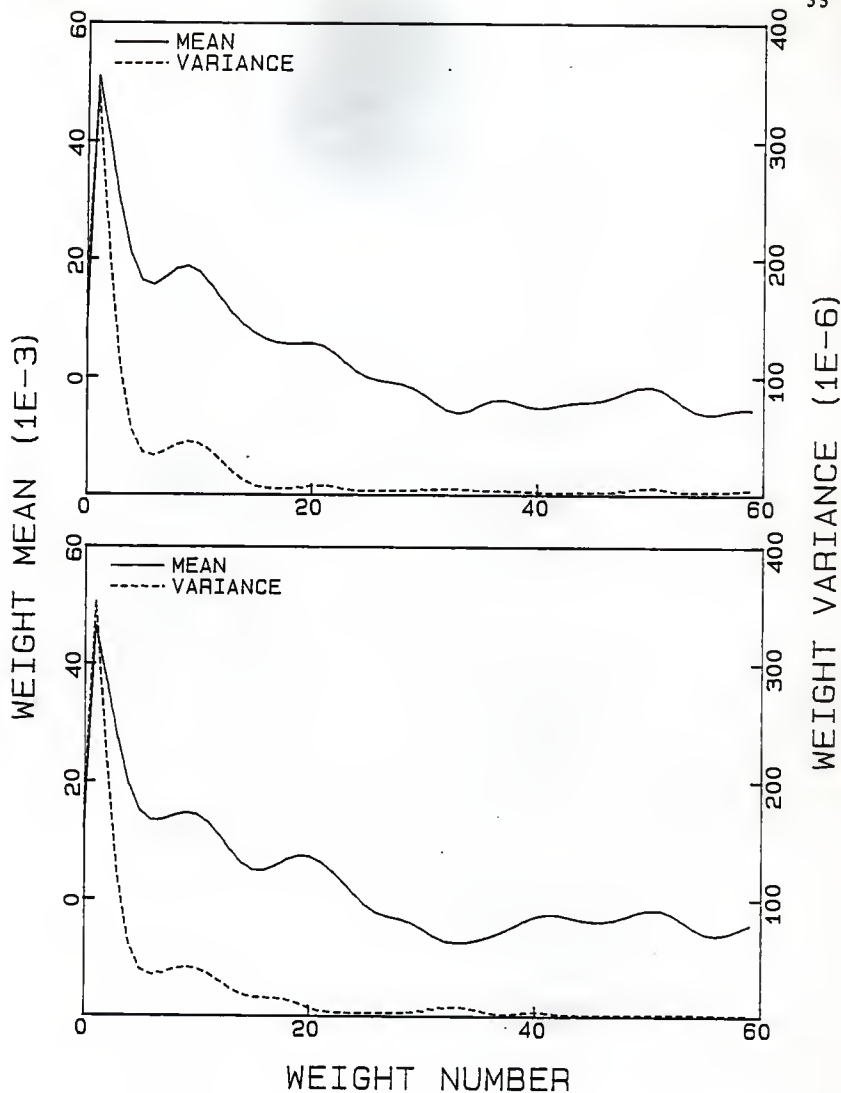
The question impending is how to extract the baud rate from this data. As previously noted, the ADF output should be the message portion of the input upon ideal cancellation. Although the cancellation is not ideal, the result is still good. By inspection of the ADF output and the corresponding input (Figures 13 and 16b) it can be concluded that the zero-crossing analysis would yield better results when performed on the ADF output than when performed on the ANC input.

The weight vector can also be used to determine the baud rate. Since the message sequence is uncorrelated from the first sample of the second bit onward, one should suspect some special

behavior at this point in the mean weight vector. By knowing the number of samples per bit to be 32, the weight number to look at would be 31 (since the delay is one). Figure 17 gives the mean and variance of the first 40 weights of two other trials produced by different runs with different noise sequences. The trend for the mean of this specific weight is for it to be a relative minimum. To distinguish it from the other relative minima of the mean weight vector, two other observations can be made. One, this minimum is the first relative minimum less than zero. Two, this minimum coincides with a maximum in the variance of the weight vector.

These criteria were not conceived through analysis but were noticed by inspecting many weight vectors for many cases. Some of the observations of this investigation should be mentioned. Recall the LMS equation (Eqn. (5.1)). If the information is to come from the weight vector it seems that the update procedure should emphasize the weight vector, i.e. make μ small. Figure 18 presents two weight vector sequences for small μ . They were calculated for the same inputs as those of Figure 17. In both comparisons between large and small μ the weight vector has the nearly the same shape. The only basic difference is the magnitude for the two cases. It seems that selection of μ for the weight vector band rate estimate, so long as stability is sustained, is not critical. This was confirmed by preliminary runs of the simulation.

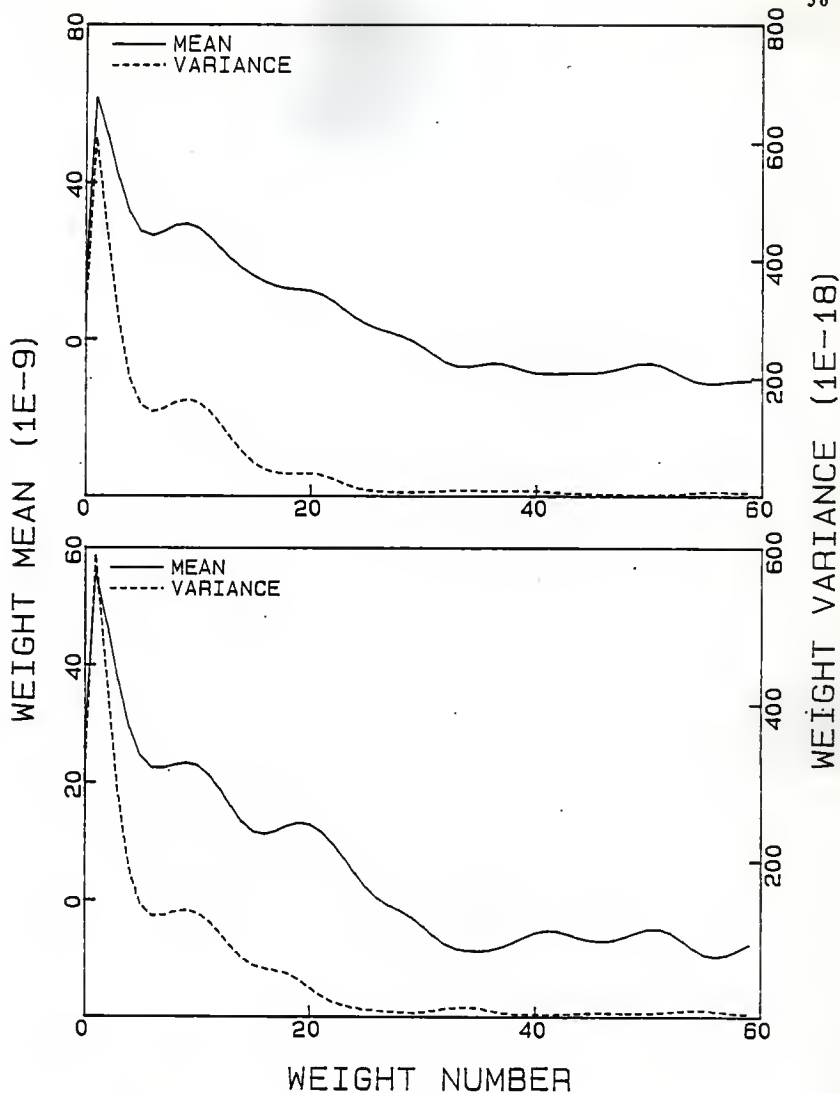
Another observation which was previously mentioned is that



$R = 300$ Hz
 $\Delta f = 300$ Hz
 $SNR = 8$ dB
 $f_s = 9600$ Hz

DELAY = 1
 $MU = 0.1E-2$
 $N = 128$
 BIAS

FIGURE 17: Baud rate estimate from the weight vector for large μ .



$R = 300$ Hz
 $\Delta f = 300$ Hz
 $SNR = 8$ dB
 $f_s = 9600$ Hz

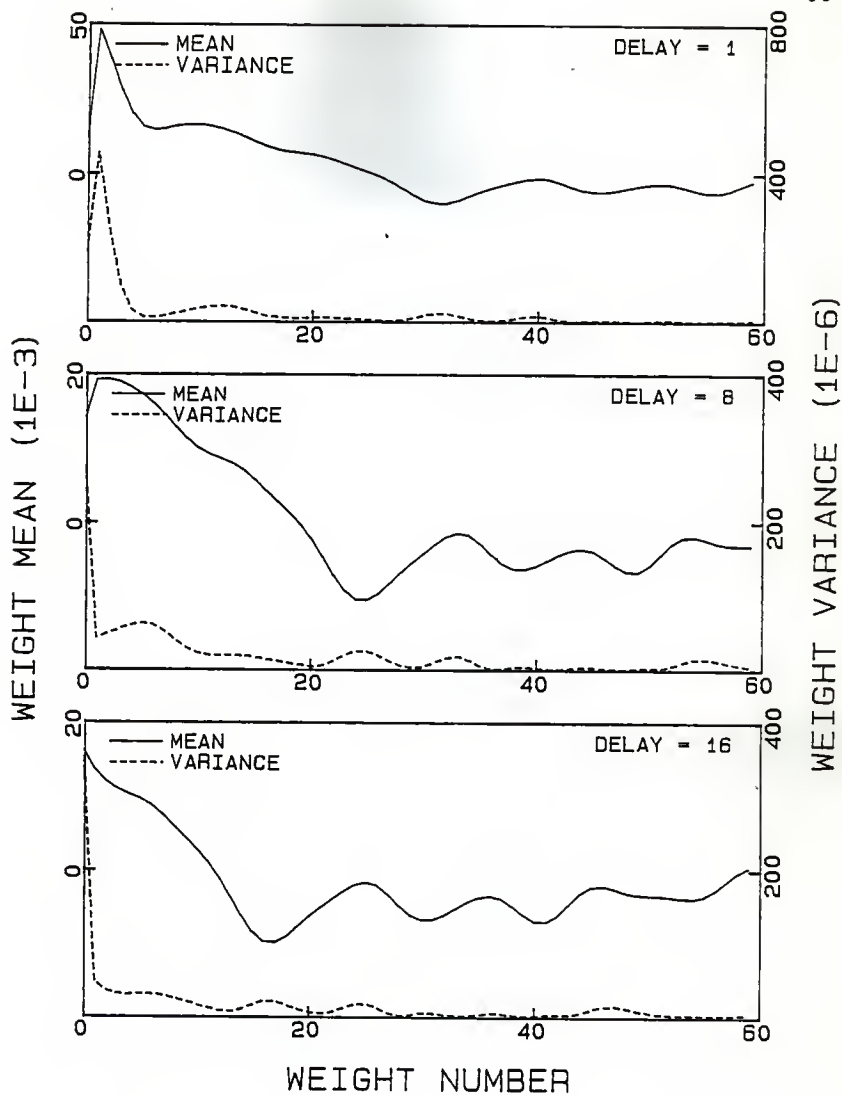
DELAY = 1
 $MU = 0.1E-8$
 $N = 128$
 BIAS

FIGURE 18: Baud rate estimate from the weight vector for small μ .

the weight vector characteristics do not change for different delays. The delay only influences the amount the weight vector is shifted with respect to the weight vector of the non-delayed condition. Supporting this observation is Figure 19 which presents three results for the same input. The only parameter which differs is the delay chosen.

One interesting observation is that when the post-filter of the DLD is omitted the weight vector mean and variance are more spurious. This makes identification of the minimum and maximum easier which should lead to better estimates. However preliminary simulation runs indicate better estimates are obtained with the post-filter included. Evidently the desirable noise reduction provided by the post-filter outweighs the undesirable smoothing effect of the filter on the weight vector.

The weight vector estimate described herein is to make estimates from the mean and variance of the individual weight vectors of each update iteration. Keep in mind that the estimate result will be expressed in number of samples therefore introducing a bias if the bit time does not equate to an integral number of sampling intervals. Another approach would be to apply the estimate procedure to each separate weight vector then average the results. This way the estimate is not limited to an integral number of sampling intervals. However preliminary runs once again suggest that better results are produced when the estimates are made from the mean and variance of the weight vectors.



$R = 300 \text{ Hz}$
 $\Delta f = 300 \text{ Hz}$
 $\text{SNR} = 8 \text{ dB}$
 $f_s = 9600 \text{ Hz}$

$\mu = 0.1\text{E-2}$
 $N = 128$
 BIAS

FIGURE 19: Effect on weight vector of varying delay.

Another attempt at estimating the band rate, which was not too successful, employed the similarities of the weight vector mean and variance with the autocorrelation of the message. Recall that a DFT of the autocorrelation of a sequence provides a power spectral density (PSD) estimate of the sequence. It is known that the PSD of the message will contain nulls at integral multiples of the band rate. The strategy is to construct a pseudo-autocorrelation sequence from either the mean or variance of the weight vector, perform a DFT, and look for the identifying nulls. Note that folding the weight vector mean about the zero axis gives a rough looking autocorrelation function. For the variance, the triangular shape is also present but with a base of one bit time instead of the expected two bit times. Once again preliminary runs discredited the credibility of this method.

Recall that one of the problems with this receiver is in setting the bandwidth of the post-filter of the DLD. Obviously if the band rate is 300 Hz and the bandwidth is 1000 Hz the band rate estimates will not be as accurate as if the bandwidth were set to 300 Hz. Thus it makes sense to try to determine the bandwidth in order to improve results. This is synonymous to determining the band rate. Therefore any of the previously mentioned methods of estimating the band rate could be used to set the bandwidth of a second low-pass filter. This narrower filtering would permit better results to be achieved. This modified system is shown in Figure 20. The intermediate filter is a Butterworth filter with 4 poles.

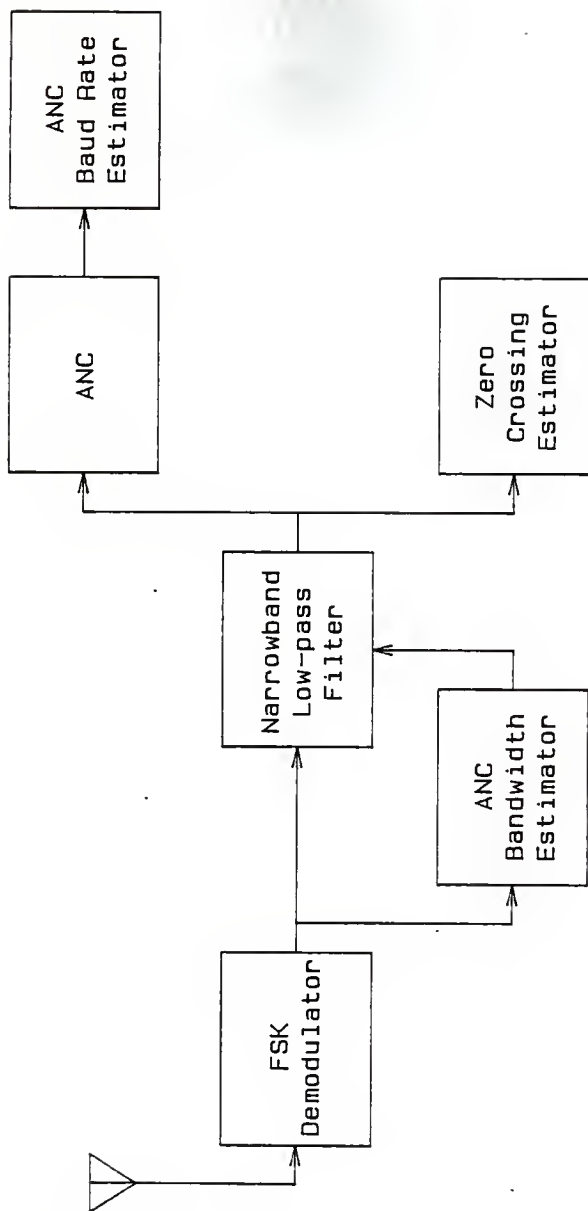


FIGURE 20: Modified system for determining FSK baud rate.

The bandwidth determination of the filter is performed using a DFT analysis of the weight vector instead of one of the bandwidth estimates previously discussed. One reason for this choice is that the bandwidth estimates are either fairly good or very bad. For example consider the estimate that is based on the minimum of the mean weight vector coinciding with the maximum of the variance of the weight vector. If the points coincide the estimate will be very accurate, if they do not the estimate will depend on if the condition is met somewhere else in the sequence.

Another reason for using the DFT technique is that it is very stable. Consider the ADF filter. Its input is the message plus noise sequence and its output is the forward shifted message sequence. The weight vector is the impulse function which accomplishes this to the best degree. As such the DFT of the weight vector provides the transfer function of this impulse function. If the task is performed well the transform should indicate the bandwidth. This bandwidth estimator might not provide the exact bandwidth every time but should be very consistent and should not fluctuate wildly.

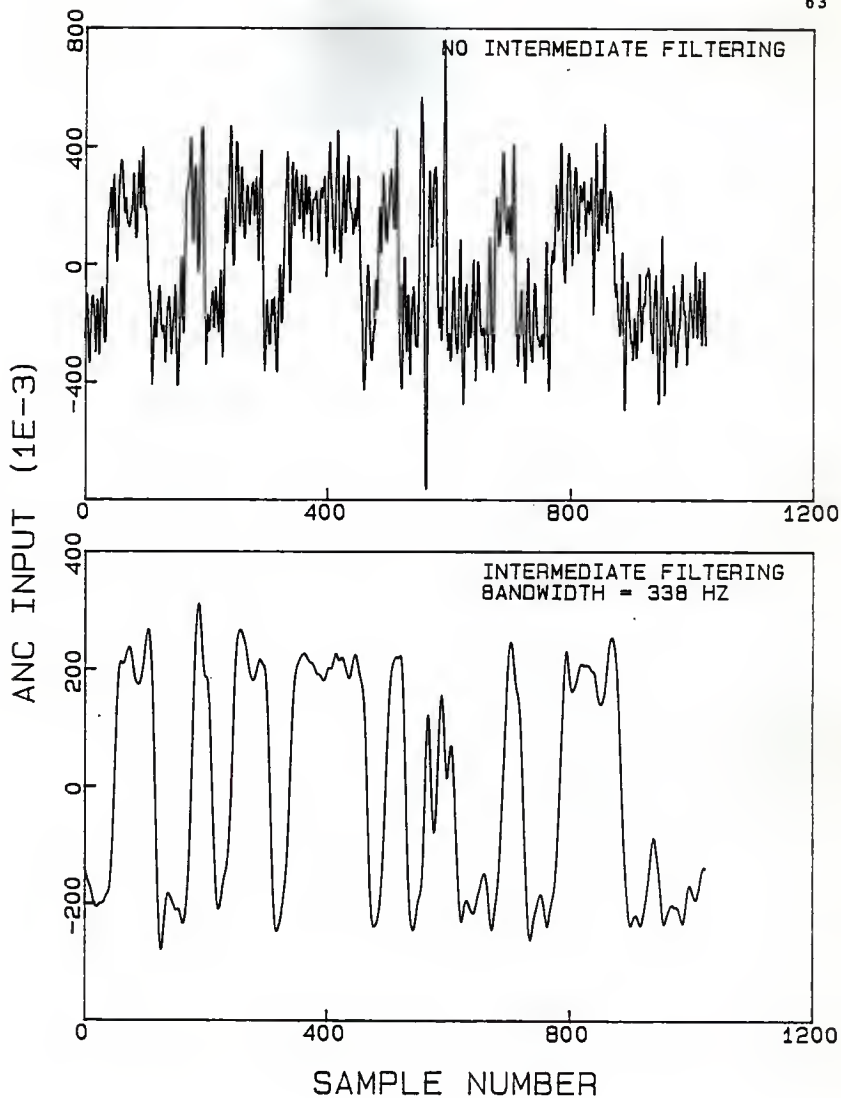
The bandwidth estimate used is twice the 3 dB bandwidth of the transform of the weight vector. This choice was arbitrarily made and found to give good results. The justification for the choice is based on the transform of a message sequence. For this transform the magnitude drops about 3 dB at one half the bandwidth which coincides with the arbitrary choice. It was found that the bandwidth estimate is more accurate when μ is small.

This is hard to explain since it was previously shown that the weight vector shape was more or less independent of the choice of μ . It could be possible that as the SNR gets lower the weight vector values become dependent upon the choice of μ . Further simulation will be needed to resolve this issue.

Figure 21 shows the input to the ANC when it is taken directly from the output of the DLD. This is the top plot. The bottom plot shows the input to the ANC after the intermediate filtering has been performed. Figure 22 shows the ANC output sequences when the input was the sequence of the bottom plot of Figure 21. It was previously indicated that the ADF output should be a clean version of the message, which it is, and that the band rate estimate could be made with a zero-crossing analysis. Unfortunately, the ADF output appears to be too heavily filtered; therefore, the zero crossings might be displaced due to the smoothing. The best results should come from using the zero-crossing analysis on the ANC input after it has been intermediately filtered. With intermediate filtering, the weight vector mean and variance of Figure 22 are beginning to look more like the autocorrelation function. Again the smoothing is undesirable in the sense that it will probably be harder to determine the band rate from the minimum of the mean and maximum of the variance.

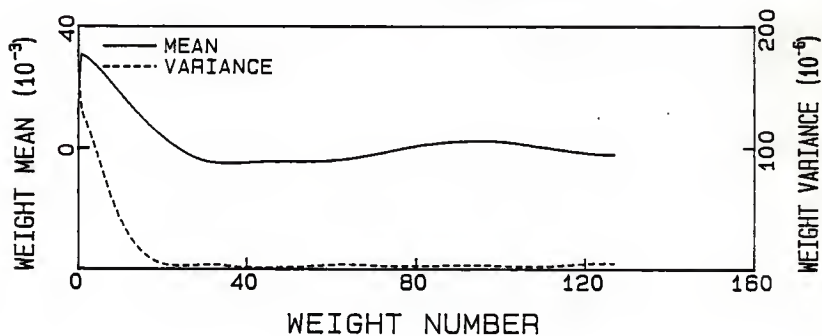
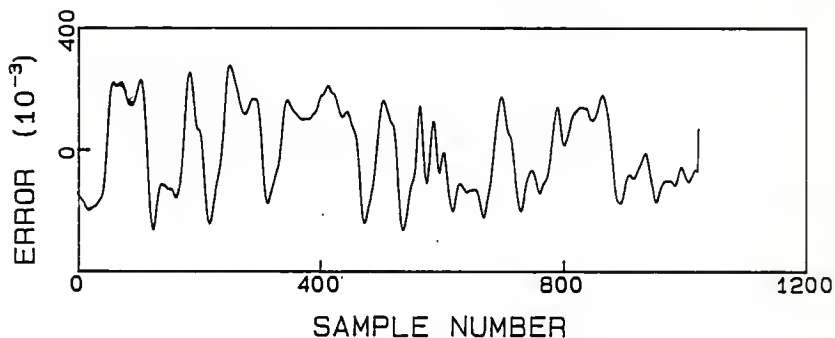
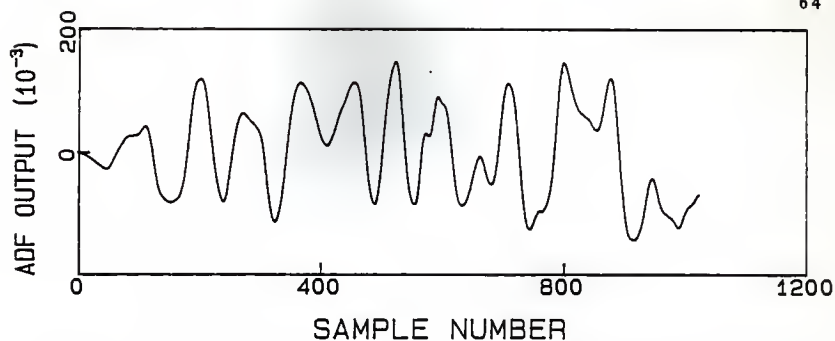
Estimator Definitions

Based on the preceding observations, there are several classes of estimators worth considering. They are listed and



$R = 300 \text{ Hz}$
 $\Delta f = 300 \text{ Hz}$
 $\text{SNR} = 8 \text{ dB}$
 $f_s = 9600 \text{ Hz}$

FIGURE 21: Effect of intermediate filtering on ANC input.



R = 300 Hz
 Δf = 300 Hz
 SNR = 8 dB
 f_s = 9600 Hz

DELAY = 1
 MU = $0.1E-2$
 N = 128
 BIAS

FIGURE 22: ANC output information with intermediate filtering with bandwidth of 338 Hz.

described in Table 1. Notice should be taken that the parameters of the two separate ANC's during the simulation are the same with the exception of μ . For the intermediate ANC (the one masking the bandwidth estimate) μ is small. A large value of μ is used for the final ANC. Therefore any estimates obtained from the intermediate ANC (i.e., any estimates made independent of the second ANC) should be associated with small μ . The estimates made with the final ANC are accordingly associated with large μ .

ACRONYM	MEANING	DEFINITION
MMI* MMF*	Min-Max	first minimum of the weight vector mean that coincides with a maximum of the vector variance
M2MI M2MF	Min-2-Max	first minimum and that a maximum is within two weights of the minimum
AMI AMF	Absolute minimum	absolute minimum of the weight vector mean
MLTOI MLTOF	Min<0	first minimum of the weight vector mean that is less than 0
MZCI MZCF	Mean<0	first weight of the mean that is less than 0
IXCF	ZC of ANC input	zero-crossing analysis of ANC input
EZCI EZCF	ZC of ANC error	zero-crossing analysis of ANC error
YZCF	ZC of ADF output	zero-crossing analysis of ANC output
MAI MAF	mean/auto-corr.	similarity of mean weight vector to the message autocorrelation
VAI VAF	var./auto-corr.	similarity of variance of vector to the message autocorrelation
*The I indicates the estimate was made from the intermediate ANC and the F indicates it was made from the final ANC.		

TABLE 1: Estimator Definition

VI. Results and Conclusions

All results presented in this section were generated with simulation runs of 100 trials each. The estimate of each trial was normalized as such to give a fractional error

$$a = \frac{\text{baud rate estimate} - \text{baud rate}}{\text{baud rate}} \quad (6.1)$$

In the simulation the baud rate was actually estimated in number of sampling intervals. The normalization was done taking this into account so the result is the same as if the estimate was made in bps. The bandwidth estimates, if normalized, were also done in this manner.

The results are presented as the mean error, $\bar{a}$, and the root-mean-square (rms) error, a_{rms} . They are the sample mean and sample standard deviation of the error. That is

$$\bar{a} = \frac{1}{N} \sum_{i=1}^N a_i \quad (6.2)$$

and

$$a_{\text{rms}} = \sqrt{\frac{1}{N} \sum_{i=1}^N (a_i - \bar{a})^2} \quad (6.3)$$

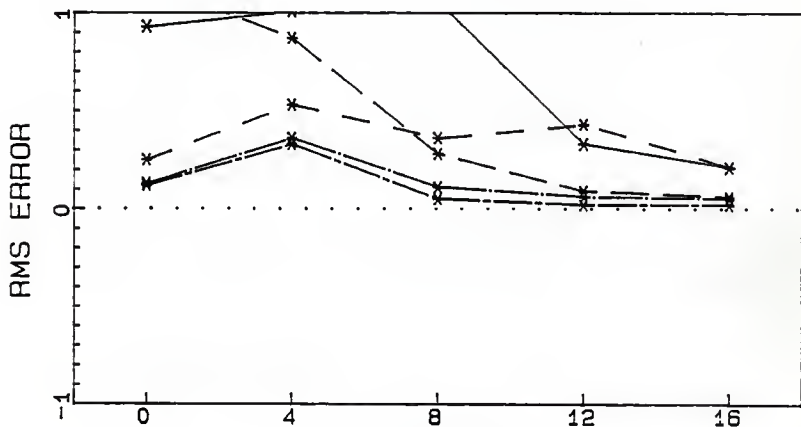
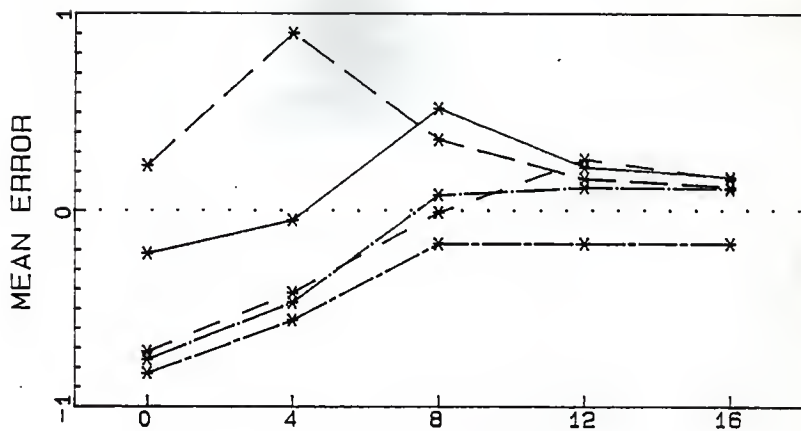
where a_i is the error of the i^{th} iteration and N is the number of trials. For $N = 100$ the sample mean and standard deviation should be a good estimate of the distribution mean and standard deviation. The results should be judged according to both the parameters since it is possible for an estimator to give a consistent but extremely biased estimate or to give a highly fluctuating estimate whose mean is within tolerable limits.

Evaluation of Estimators

The results using the estimators defined in the previous section are presented by estimator class in Figures 23a-d. For the weight vector estimators without intermediate filtering shown in Figure 23a, only two estimators give good results down to 8 dB. They are the MLTOI and the MZCI estimators. The others are too inconsistent to be of any good. When the intermediate filtering is incorporated, the overall results for this group of estimators do not get much better. This time the good estimators, as seen in Figure 23b, are the AMF and the MLTOF estimators. The filtering has introduced a distinct bias in the MZCF estimator and the rms error for the others are intolerable.

In Figure 23c the results for the zero crossing analysis using the ANC data are presented. Obviously, any estimator should work fairly well for larger SNR, e.g., 12 dB. Thus, all the estimators except the XZCF estimator should be ruled out. The XZCF estimator gives the best results for SNR of 12 and 16 dB of any estimator considered thus far. Its performance for SNR = 8 dB is slightly worse than the MLTOI estimator.

Figure 23d presents the pseudo-correlation sub-class of the weight vector class. The rms error for this group is the best of any group considered. However, the mean error discredits every one of the estimators. Note the VAI estimator. It is very consistent and has a steady mean error. It would be very useful if the bias could be directly linked to the baud rate. Until this connection can be made it should be considered as a poor



SNR (dB)

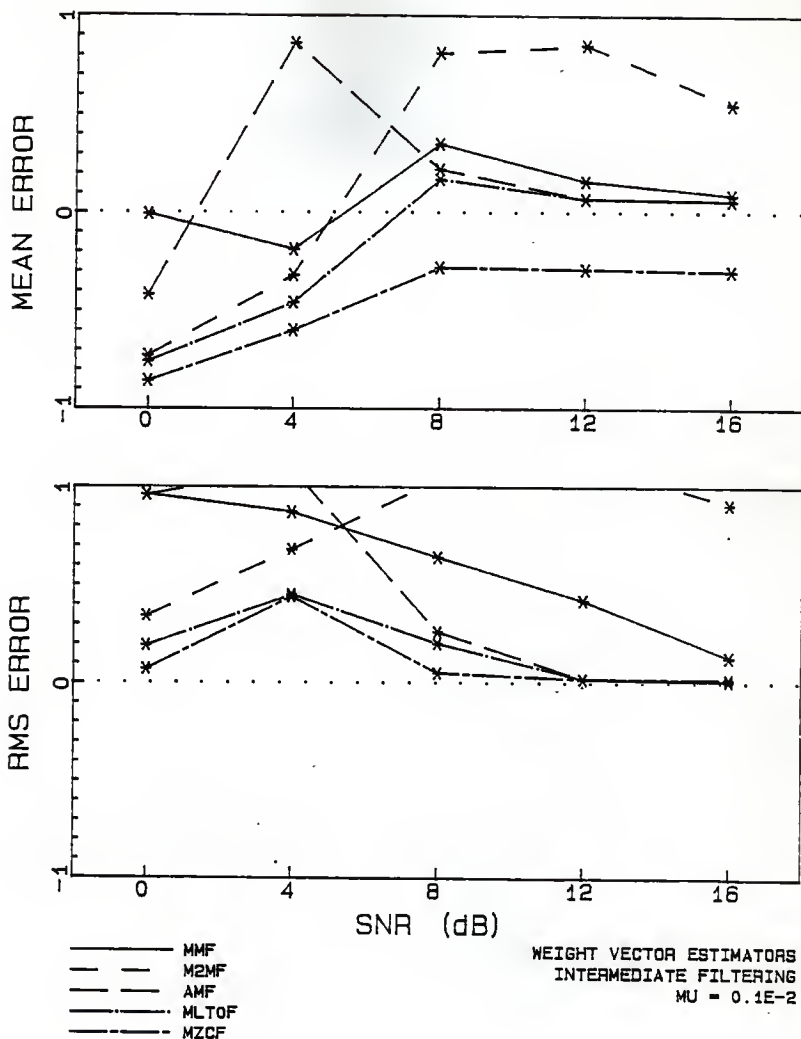
WEIGHT VECTOR ESTIMATORS
NO INTERMEDIATE FILTERING
MU = 0.1E-8

MMI
M2MI
AMI
MLTOI
MZCI

R = 300 Hz
 Δf = 300 Hz
 f_s = 9600 Hz

DELAY = 1
N = 128
BIAS

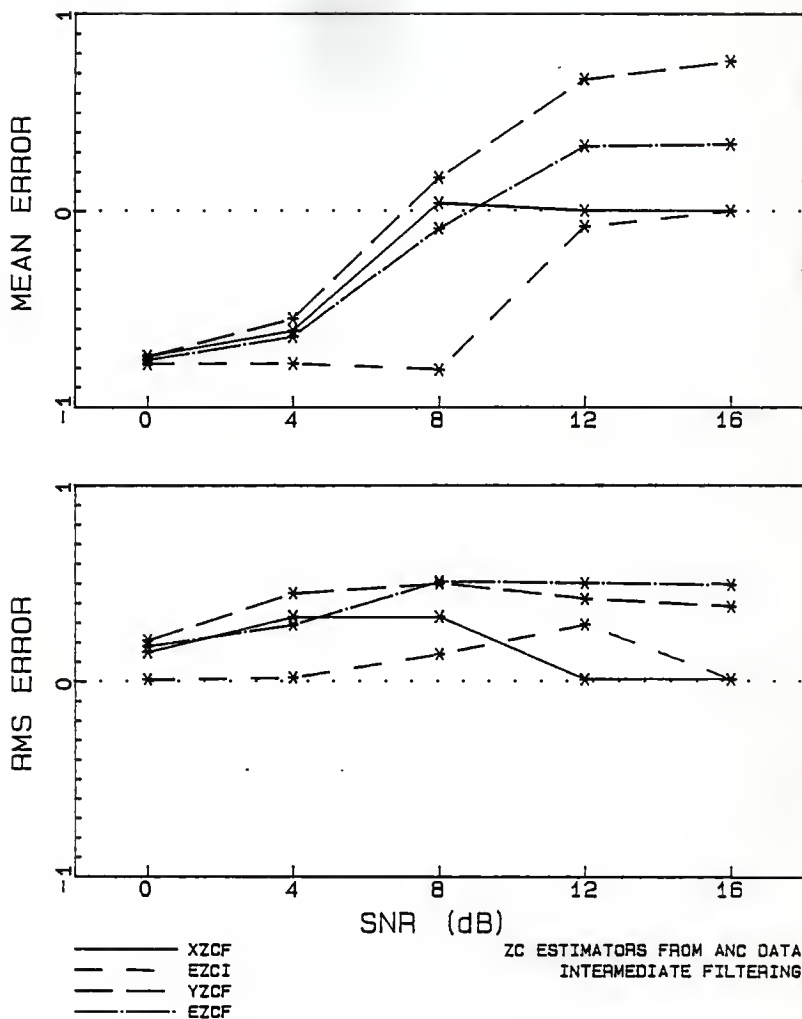
FIGURE 23a: Evaluation of estimators.



$R = 300$ Hz
 $\Delta f = 300$ Hz
 $f_s = 9600$ Hz

DELAY = 1
 $N = 128$
 BIAS

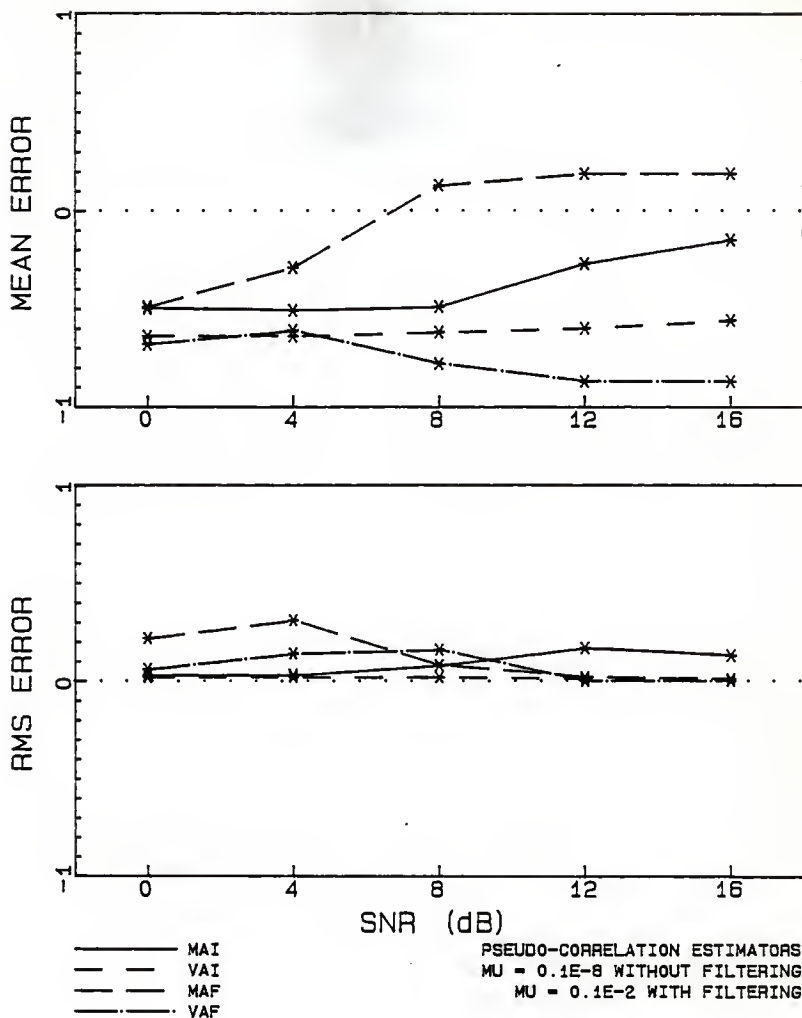
FIGURE 23b: Evaluation of estimators.



$R = 300$ Hz
 $\Delta f = 300$ Hz
 $f_s = 9600$ Hz

DELAY = 1
 $\mu = 0.1E-8$
 $N = 128$
 BIAS

FIGURE 23c: Evaluation of estimators.



R = 300 Hz
 Δf = 300 Hz
 f_s = 9600 Hz

DELAY = 1
 N = 128
 8IAS

FIGURE 23d: Evaluation of estimators.

estimator.

Demodulator Evaluation

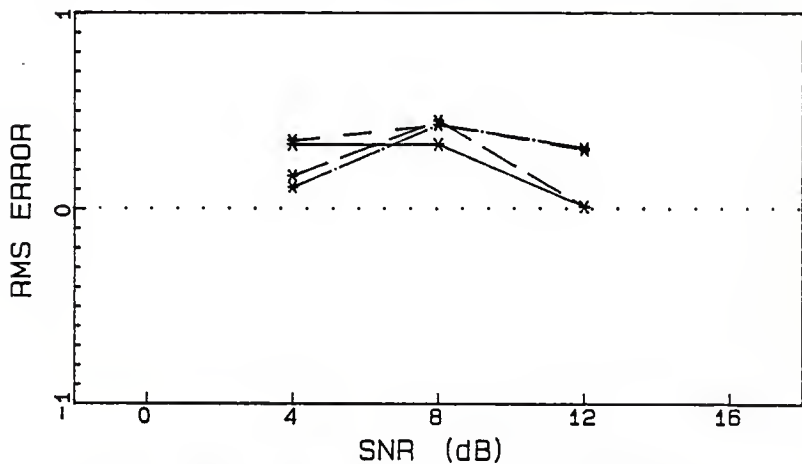
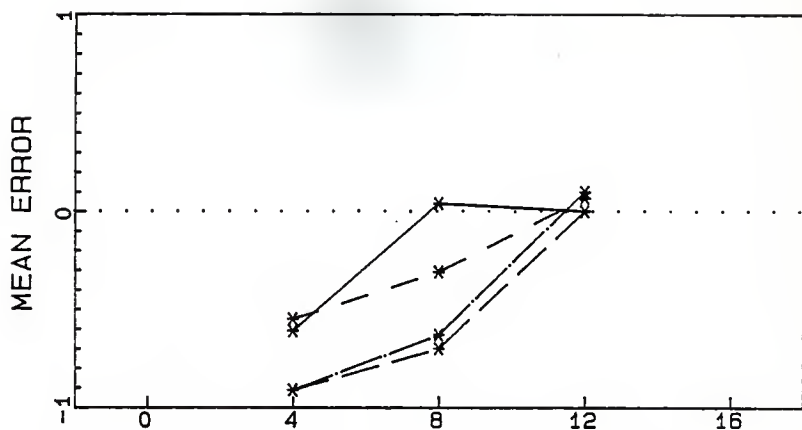
The limiter and the post-filter were included in the FSK demodulator when the previous estimator results were obtained. It was previously argued that this combination would provide the best results. In an effort to substantiate this, the different combinations of post-filter and limiter were used in obtaining the estimates. Figure 24a shows the results for the XZCF estimator. Clearly the post-filter should be included. The presence of the limiter also seems to improve the results.

Another way to evaluate the post-filter and limiter is to observe the accuracy of the bandwidth estimate when the post-filter and limiter combination is varied. Figure 24b shows these accuracy comparisons. Obviously the post-filter and limiter should be included.

Since the XZCF estimator is critically dependent on the noise reduction performed by the intermediate filter, it is also dependent on the bandwidth estimate. The previous comparisons will be made for the MLTOI estimator. This estimator does not depend on the intermediate filtering. Figure 24c gives the error results for this estimator under the same situations. Once again the results support inclusion of the post-filter and limiter.

ANC Parameter Evaluation

Another parameter set used on the basis of assumption and intuitive feeling is the ANC parameter set. It was previously argued that the delay should be 1 and the gradient constant, μ



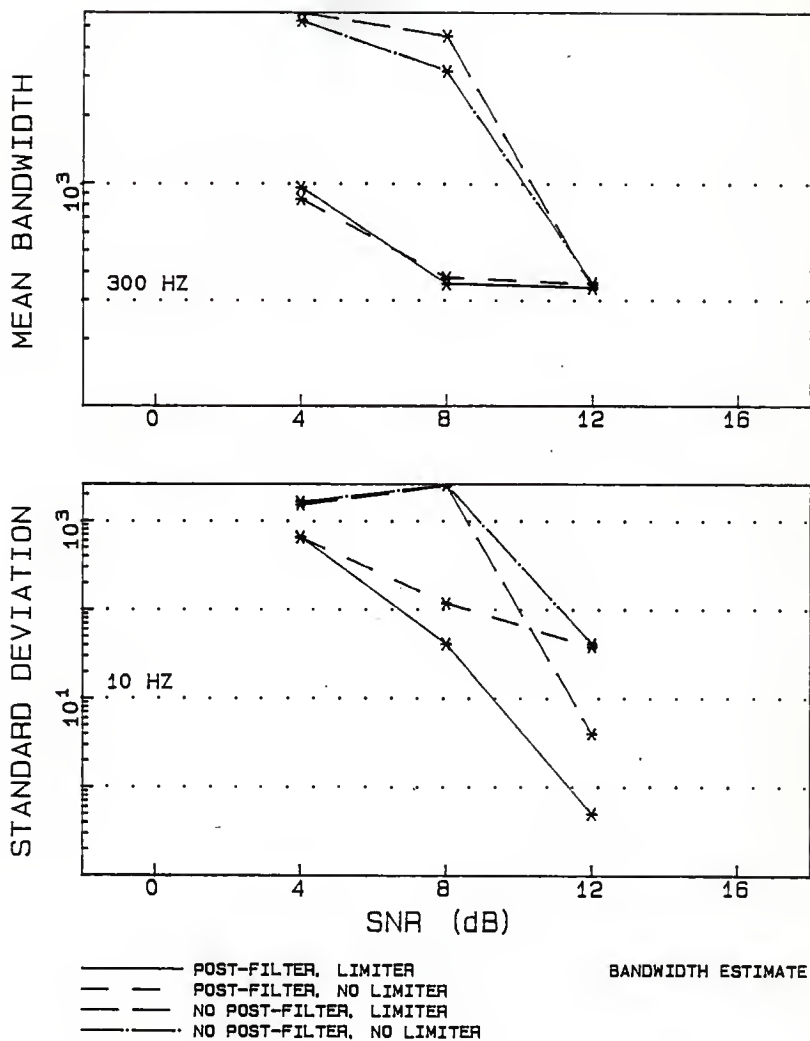
— POST-FILTER, LIMITER
 - - POST-FILTER, NO LIMITER
 — NO POST-FILTER, LIMITER
 - - NO POST-FILTER, NO LIMITER

XZCF ESTIMATOR
 INTERMEDIATE FILTERING

$R = 300$ Hz
 $\Delta f = 300$ Hz
 $f_s = 9600$ Hz

DELAY = 1
 $MU = 0.1E-8$
 $N = 128$
 BIAS

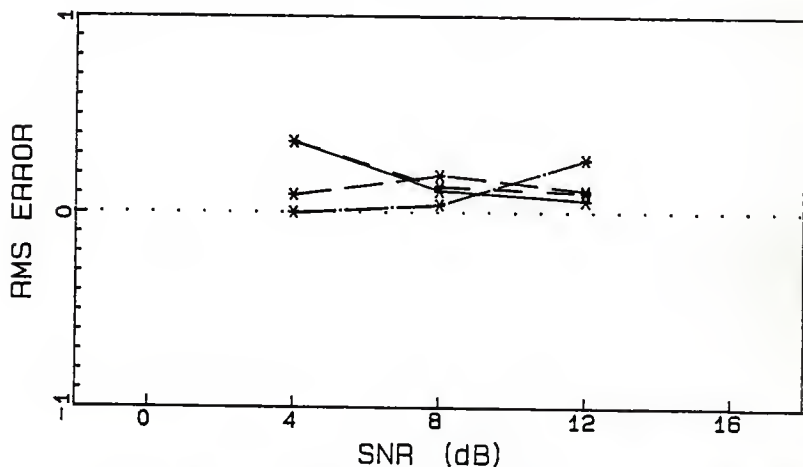
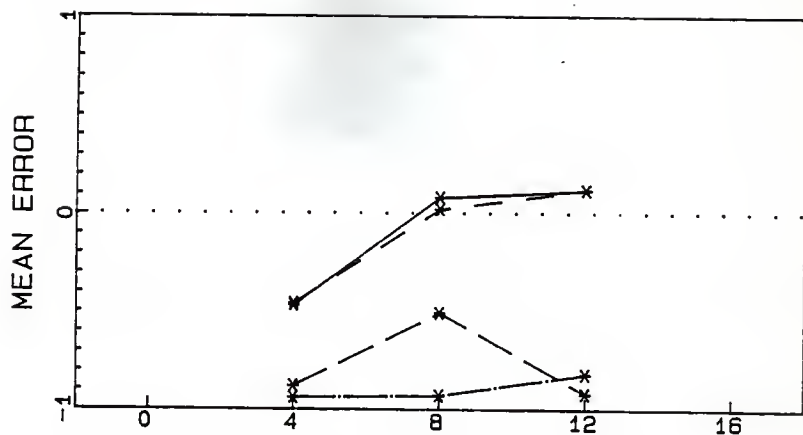
FIGURE 24a: Evaluation of FSK demodulator.



$R = 300 \text{ Hz}$
 $\Delta f = 300 \text{ Hz}$
 $f_s = 9600 \text{ Hz}$

DELAY = 1
 $MU = 0.1E-8$
 $N = 128$
 BIAS

FIGURE 24b: Evaluation of FSK demodulator.



— POST-FILTER, LIMITER
 - - POST-FILTER, NO LIMITER
 — NO POST-FILTER, LIMITER
 - - NO POST-FILTER, NO LIMITER

MLTOI ESTIMATOR
 NO INTERMEDIATE FILTERING

$R = 300$ Hz
 $\Delta f = 300$ Hz
 $f_s = 9600$ Hz

DELAY = 1
 $MU = 0.1E-8$
 $N = 128$
 BIAS

FIGURE 24c: Evaluation of FSK demodulator.

should be large. Table 2 presents the errors for a variety of combinations of these two parameters for a SNR of 8 dB. The frequency deviation is 300 Hz and the baud rate is 300 bps. The weight vector length in all cases is 128. This might be unnecessarily long but will not be detrimental. The weight vector bias has been included since it, neither, will degrade the results. The estimates were made with the ANC following the intermediate filter. The ANC making the bandwidth estimate had μ set to $0.1E-8$ and the delay as specified in the table. The resulting estimate of filter bandwidth is specified in the table.

ANC Parameters	Estimator					
	MLTOF		MZCF		Bandwidth	
	$\bar{e}$	e_{rms}	$\bar{e}$	e_{rms}	B	var
$\Delta = 1$ $\mu = 0.1E-2$	0.17	0.20	-0.28	0.05	352	41
$\Delta = 1$ $\mu = 0.1E-8$	0.41	0.21	-0.15	0.07		
$\Delta = 8$ $\mu = 0.1E-2$	0.17	0.16	-0.25	0.05	385	76
$\Delta = 8$ $\mu = 0.1E-8$	0.44	0.37	-0.06	0.47		

Table 2: Evaluation of ANC parameters

The reason for using these two estimators is that they are the better of the class as noted in the estimator evaluation. For both estimators the different delays give nearly the same results when μ is large. When it is small, the only distinct

difference is that the rms error is worse for the delay of 8 for the MZCF estimator. As for the bandwidth estimate, the delay should be 1 in order to obtain the better estimate.

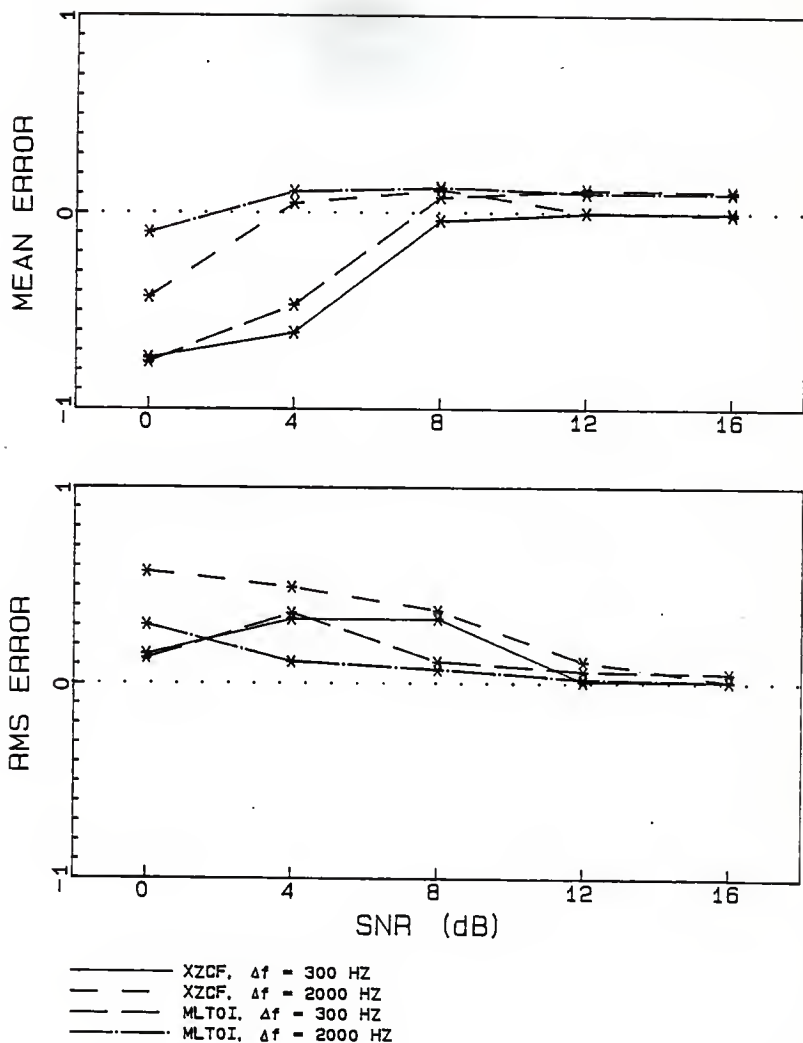
The bandwidth estimate gives better results for the shorter delay. Hence the delay is best set when it is one. This will also eliminate the possible loss of correlation when the baud rate is large. The problem results from the delay extending past the last bit difference for which correlation exists.

One explanation for the similarities of results is that the gradient constant might not have been large enough to make a distinct difference. Perhaps it can be made larger by sacrificing weight vector length in an attempt for better results. As for a decision, μ will be left at the large value of $0.1E-2$ in keeping with the philosophy that this will provide better cancellation.

Varying Frequency Deviation

All of the previous results have been generated for a frequency deviation of 300 Hz. This is the worst case possible since the baud rate is also 300 bps. Therefore, results should improve when the deviation is increased and the baud rate is kept constant. Figures 25a and 25b substantiate this conclusion.

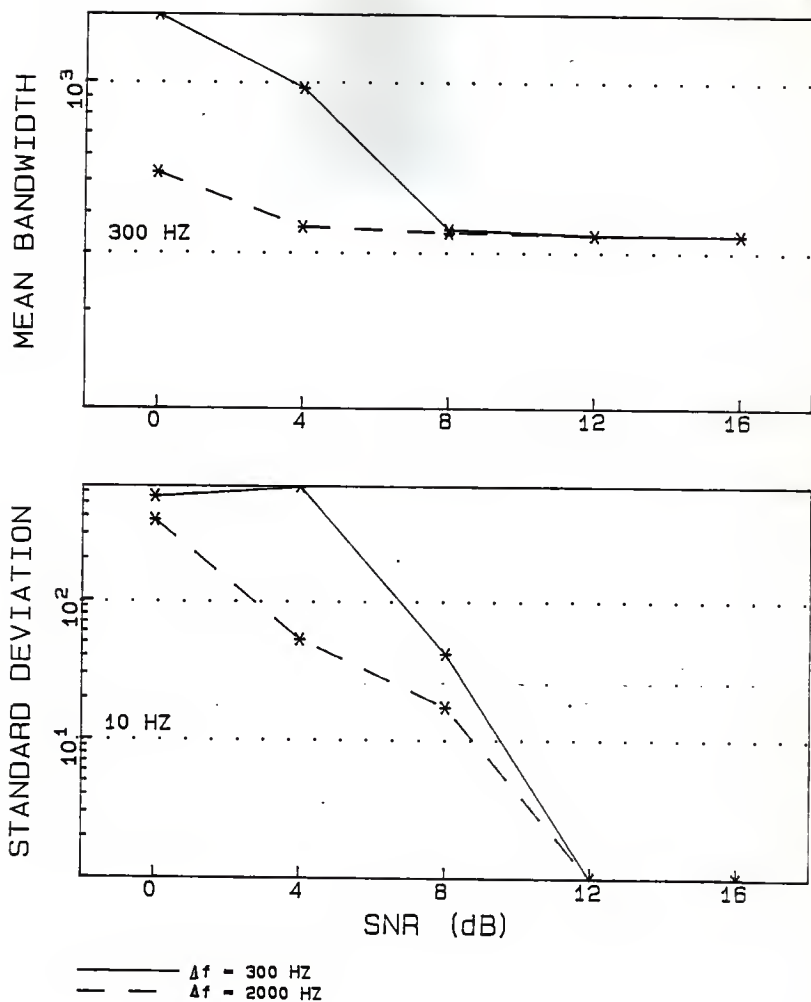
In Figure 25a, the results for the XZCF estimator and the MLTOI estimator are presented for the cases of Δf equal to 300 Hz and 2000 Hz. According to the mean error, both estimators do a better job for the larger deviation at lower SNR. However, the rms error has become slightly worse for the XZCF estimator and



$R = 300$ Hz
 $f_s = 9600$ Hz

DELAY = 1
 $\mu = 0.1E-8$
 $N = 128$
 BIAS

FIGURE 25a: Effect of varying frequency deviation of the FSK signal on the estimators.



R = 300 Hz
 $f_s = 9600$ Hz

DELAY = 1
 MU = $0.1E-8$
 N = 128
 BIAS

FIGURE 25b: Effect of varying frequency deviation of the FSK signal on the bandwidth.

slightly better for the MLTOI estimator. This unsuspected degradation for the XZCF estimator is unexplainable.

Figure 25b shows how the bandwidth estimate is affected by the frequency deviation of the FSK signal. The estimate is substantially better for the larger deviation.

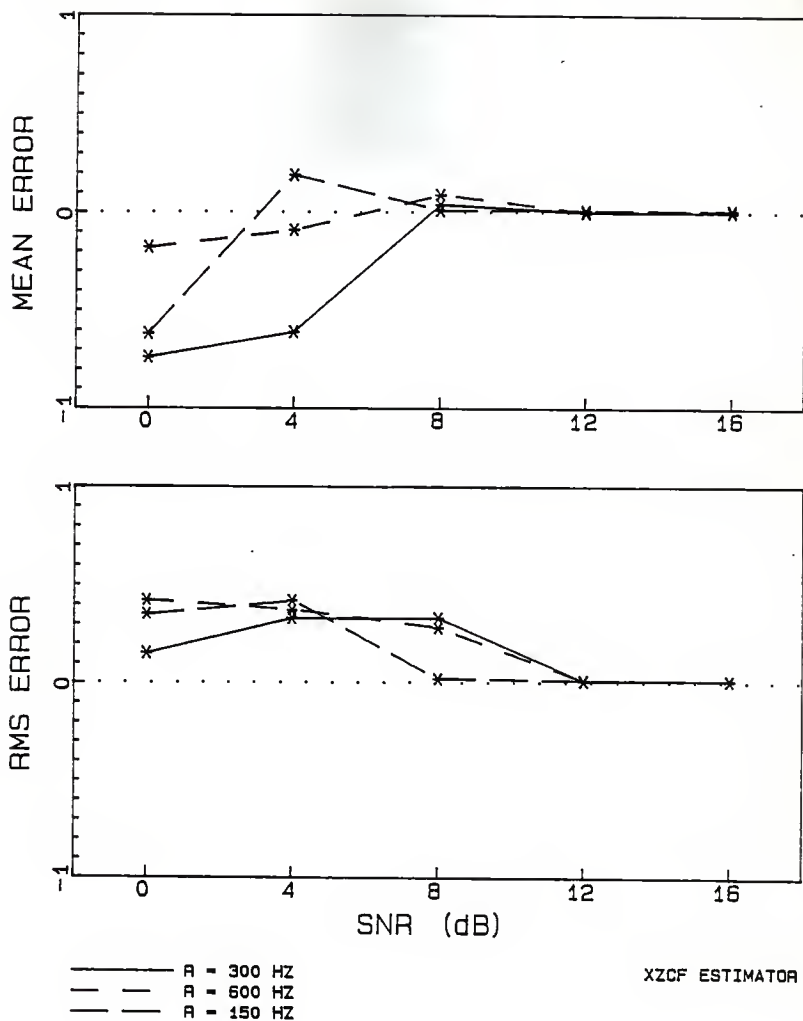
Different Baud Rates

As stated in the introduction the design should work for baud rates up to 1000 bps. Therefore, the results of the estimators at different baud rates should be evaluated. Figures 26a-e give the results of this evaluation.

The estimates of the XZCF estimator made for three different baud rates are plotted in Figure 26a. The three baud rates are 150, 300, and 600 bps. Since the deviation must be at least as large as the baud rate, it was set to 2000 Hz for each case. The conclusion to make is that the XZCF estimator performs well for a variety of baud rates down to a SNR of 8 dB. The 150 bps baud rate has a significantly smaller rms error at 8 dB which perhaps indicates the estimator will work better for lower baud rates. This is agreeable with the manner that the zero crossing analysis is performed and with the way that errors may be introduced.

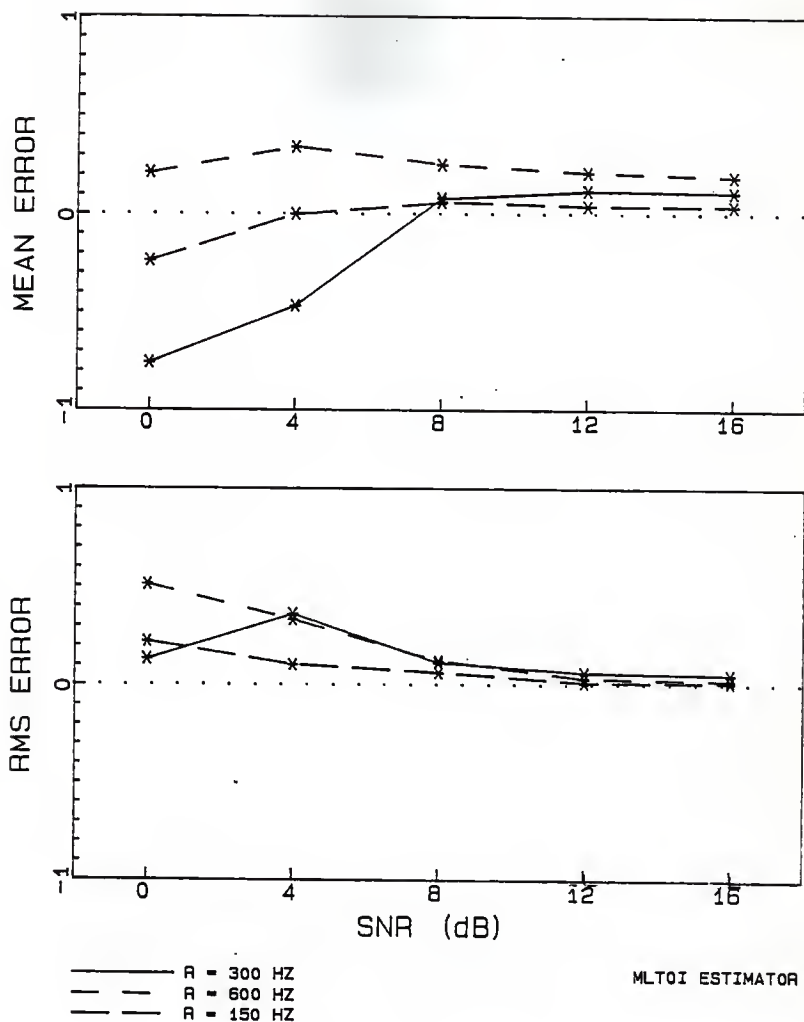
The MLTOI estimator also performs slightly better at lower SNR for the lower baud rates as evidenced by Figure 26b. There is a trend in the mean error to have an increasing bias as the baud rate increases. The rms error follows a similar increasing pattern although not as severely as the mean error.

One explanation for the estimators performing better at



$\Delta f = 2000$ Hz
 $f_s = 9600$ Hz

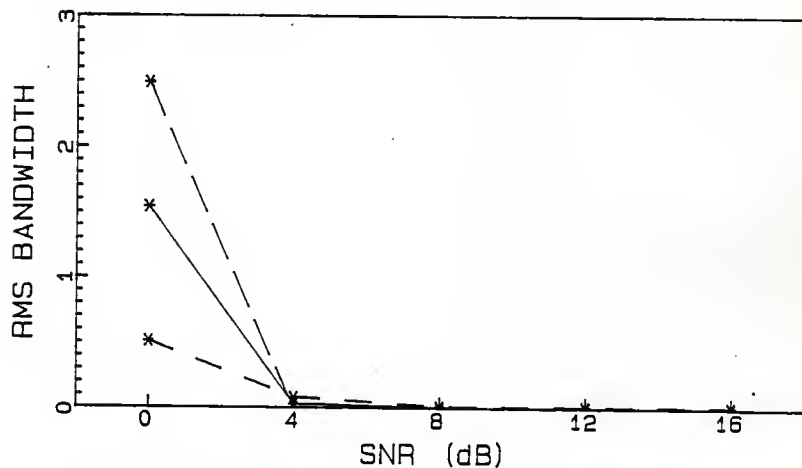
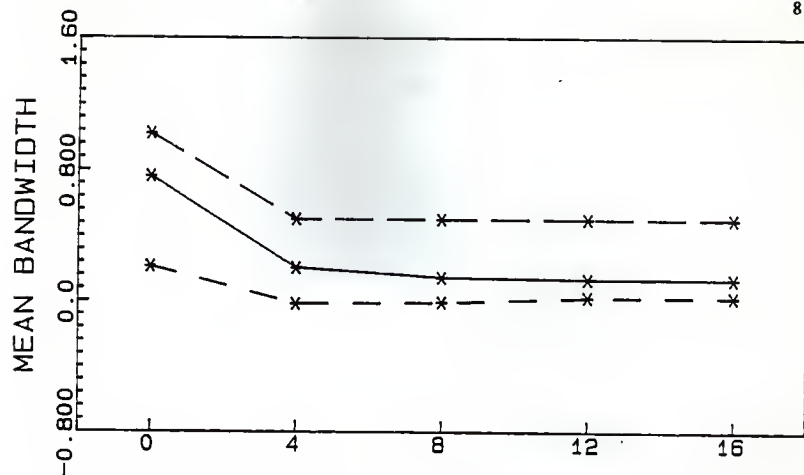
FIGURE 26a: Effect of varying the baud rate on the XZCF estimator.



$\Delta f = 2000$ Hz
 $f_s = 9600$ Hz

DELAY = 1
 MU = 0.1E-8
 N = 128
 BIAS

FIGURE 26b: Effect of varying the baud rate on the MLTOI estimator.



——— $R = 300$ HZ
 - - - $R = 500$ HZ
 - . - $R = 150$ HZ

BANDWIDTH NORMALIZED
TO THE BAUD RATE

$\Delta f = 2000$ HZ
 $f_s = 9500$ HZ

DELAY = 1
 MU = 0.1E-8
 N = 128
 BIAS

FIGURE 26c: Effect of varying the baud rate on the bandwidth estimate.

lower baud rates is that the sampling frequency is too low. If the sampling frequency is larger, more samples per bit would be observed. This would have a direct effect on any estimator from the weight vector class of which the MLTOI estimator is a member. It would also affect the zero crossing class.

The results of the bandwidth estimate are shown in Figure 26c. The tendency this time is for the bias to increase as the band rate decreases. This shows that the bandwidth estimate made in this way would not be a good baud rate estimate. However, the purpose of the intermediate filter is to reduce the noise power which the post-filter of the DLD let pass. The fluctuating bias should not matter since in all cases the resulting bandwidth is a significant improvement over the 1000 Hz bandwidth of the post-filter, e.g., the bandwidth estimate for the 150 bps band rate case is 225 Hz. The best feature of the bandwidth estimator is that it is very consistent. As long as the SNR is 4 dB or more, the intermediate filter will reduce noise power in a consistent manner.

Evaluation Against the Traditional Method

The traditional method for determining the band rate of an FSK signal is the use of zero crossing methods. Therefore the proposed method should be compared to this traditional method to determine if the proposed method has merit. Note that the only difference between the XZCF estimator and the traditional method is the inclusion of an intermediate filter to further reduce the noise power. The bandwidth of the filter is set using the ANC.

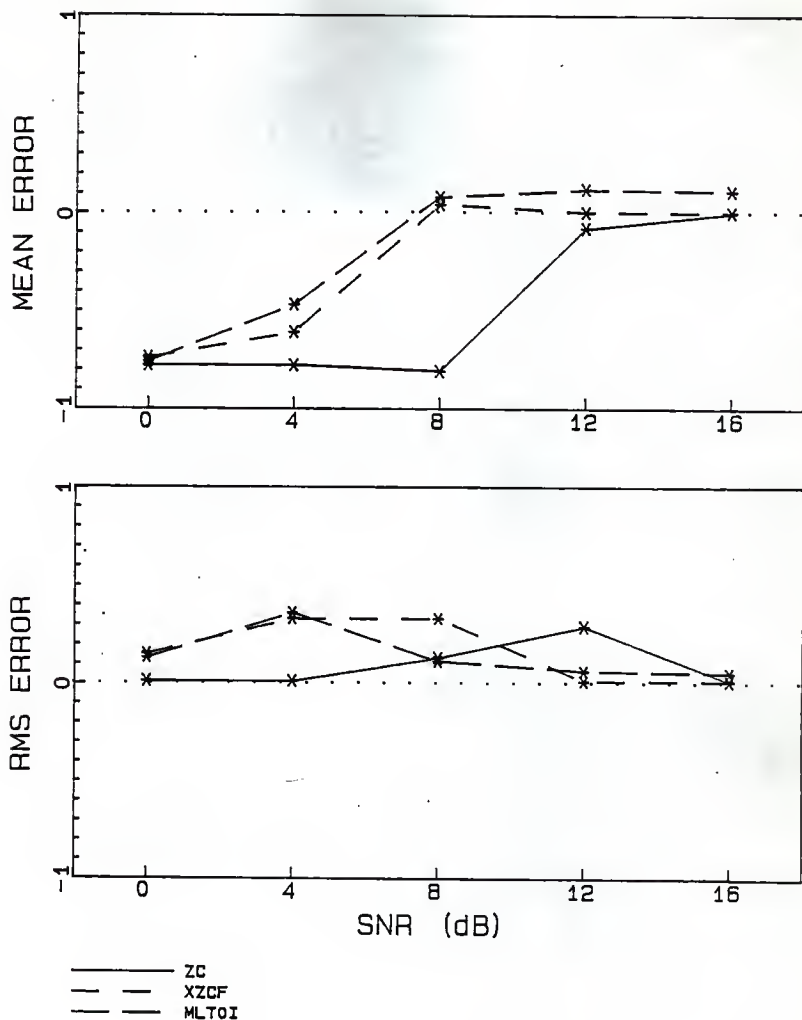
In order to make the judgement, the results of these two methods along with the results using the MLTOI estimator are plotted in Figures 27e-d for various bend rates and signal deviations.

In Figure 27e the bend rate is 300 bps and the deviation is 300 Hz. The accuracy of the traditional method drops somewhere above 12 dB while both of the proposed methods are good down to nearly 8 dB.

The bend rate in Figure 27b is 600 bps and the deviation is 2000 Hz. This time the MLTOI estimator is the poorest estimator. The ZC and the XZCF estimator give nearly identical results. This could be for two reasons. One, the addition of the intermediate filter at about 600 Hz bandwidth did not make a significant reduction in noise power. Two, the larger bend rate means more samples per bit time, and thus, a steadier zero crossing analysis can be made.

When the bend rate is decreased to 150 bps with the same deviation, as in Figure 27c, the XZCF estimator does indeed perform better than the traditional ZC estimator. However, the MLTOI estimator performs the best of the three. Its results are good to 4 dB.

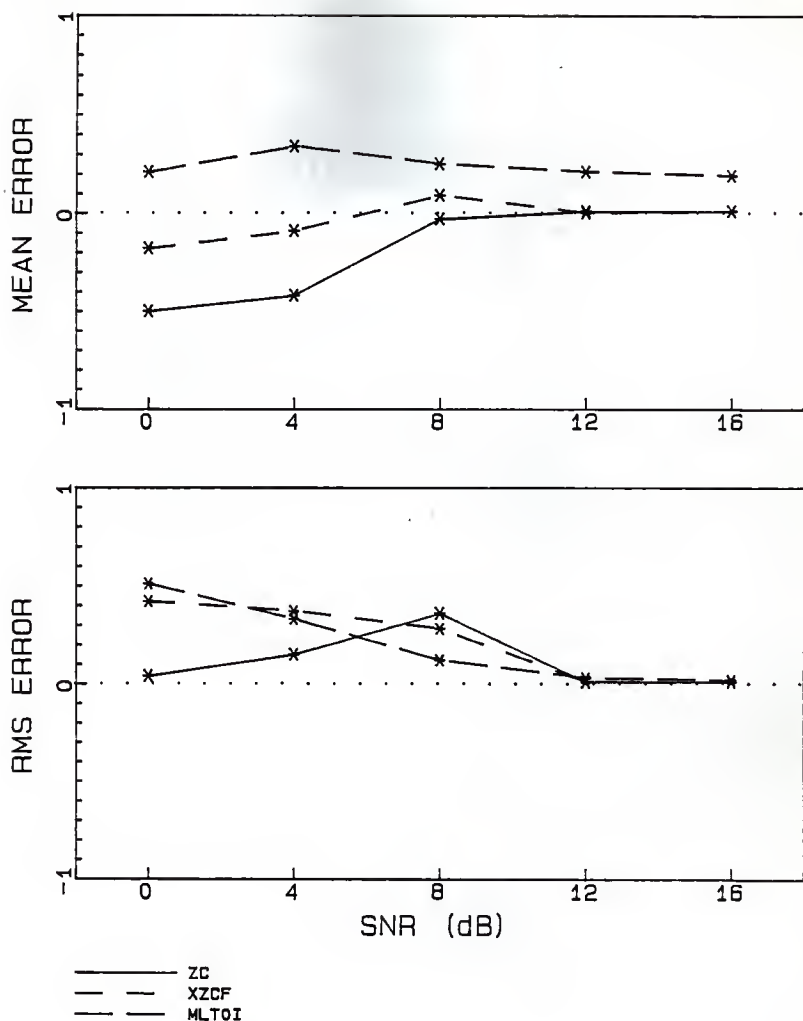
Figure 27d prompts the same conclusions for a bend rate of 300 bps and a signal deviation of 2000 Hz. This time, however, the MLTOI estimator contains a bias which is unexplainable. Still, its estimates are relative accurate to 4 dB. Again, the traditional ZC method is useless below 12 dB.



$R = 300$ Hz
 $\Delta f = 300$ Hz
 $f_s = 9600$ Hz

DELAY = 1
 $MU = 0.1E-8$
 $N = 128$
 BIAS

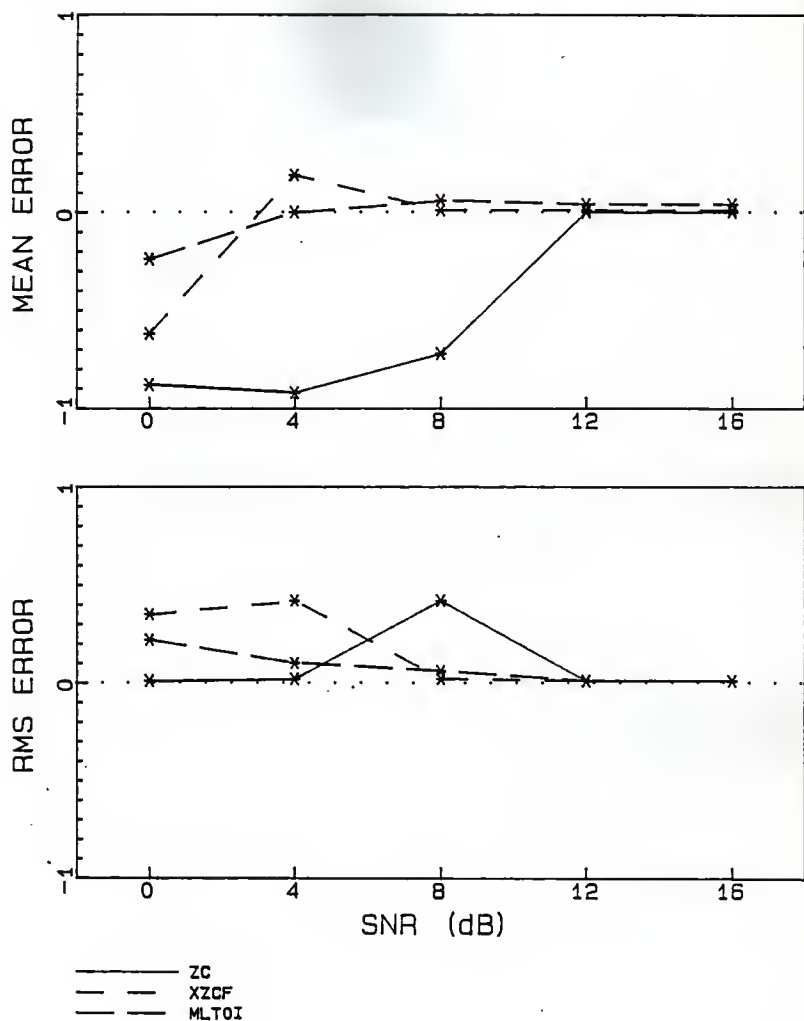
FIGURE 27a: Evaluation of proposed estimators against the traditional zero-crossing estimator



$R = 600$ Hz
 $\Delta f = 2000$ Hz
 $f_s = 9600$ Hz

DELAY = 1
 $MU = 0.1E-8$
 $N = 128$
 BIAS

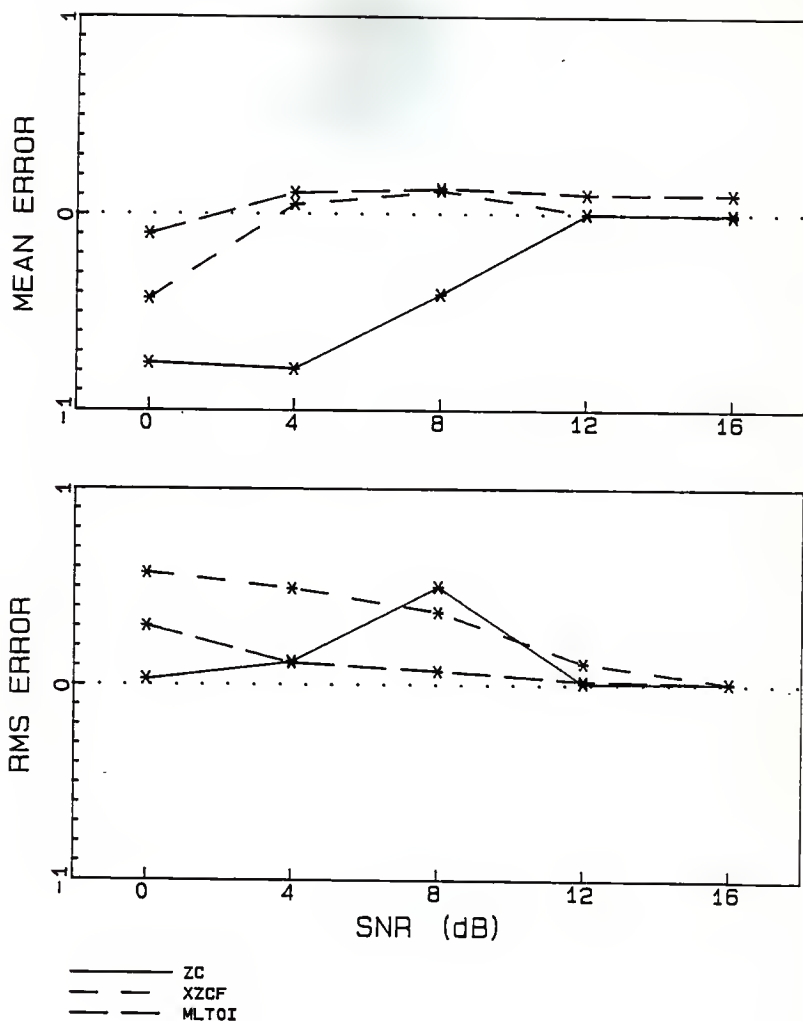
FIGURE 27b: Evaluation of proposed estimators against the traditional zero-crossing estimator.



$R = 150$ Hz
 $\Delta f = 2000$ Hz
 $f_s = 9600$ Hz

DELAY = 1
 $MU = 0.1E-8$
 $N = 128$
 8IAS

FIGURE 27c: Evaluation of proposed estimators against the traditional zero-crossing estimator.



$R = 300$ Hz
 $\Delta f = 2000$ Hz
 $f_s = 9600$ Hz

DELAY = 1
 $\mu = 0.1E-8$
 $N = 128$
 BIAS

FIGURE 27d: Evaluation of proposed estimators against the traditional zero-crossing estimator.

Conclusions

Two classes of estimators proposed provide better results than the traditional zero crossing method. Both methods rely on the ANC, one directly and the other indirectly.

The weight vector class of estimators uses the mean and variance of the weight vectors of the iterative convergence procedure to obtain the baud rate. Of the estimators of this class, the MLTOI estimator was found to be one of the better estimators. Figure 28 shows how the MLTOI estimate is acquired. The estimate defines the bit time in number of sampling intervals as being the delay of the ANC plus the number of the weight which represents the first relative minimum of the weight vector which is less than zero. The bit interval is then obtained from the weight number plus delay by multiplying by the sampling interval. Then the baud rate is found by inverting the bit interval. The estimator gives good results over a large part of the desired range of baud rates and frequency deviations. Its estimates are reliable down to 8 dB in most instances. This estimate performs better than the traditional method for nearly all baud rates and appreciably better for low baud rates.

The other class of estimators relies indirectly on the ANC. This time the ANC is used to obtain a bandwidth estimate of the message. Then intermediate filtering is performed using the bandwidth estimate to further reduce the noise power. The baud rate estimate is then obtained using zero crossing analysis just as the traditional method does. The analysis is performed on

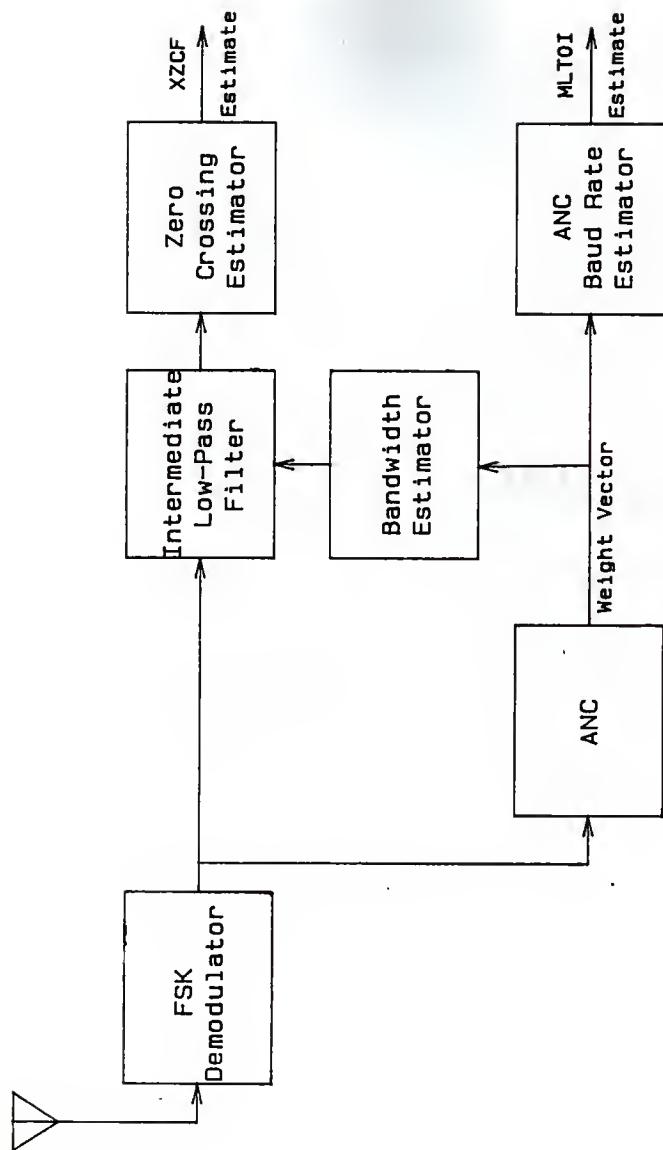


FIGURE 28: Final system for determining FSK baud rate.

either the filtered ANC input or on the various outputs from either one of the two ANC's. The best results are obtained for the XZCF estimator which is when the zero crossing analysis is performed on the intermediate filter output. This is illustrated in Figure 28. Obviously, this estimate will perform at least as well as the traditional method. When the band rate drops to the lower end of the range, this estimator performs much better than the traditional method. This is a direct result of the additional noise power cancellation attributed to good bandwidth estimation by the ANC. However, for the upper range of baud rates and the lower range of frequency deviations, this estimator will degrade just as fast as the traditional method when the SNR is decreased.

The bandwidth estimation provided by the ANC should be considered as a significant achievement. The estimate rarely underestimates the bandwidth and the accuracy remains stable down to 4 dB SNR. Obviously this addition to the system is only valuable for the surveillance situation since the bandwidth would be known in other applications.

One factor which definitely influences the estimators is the sampling frequency. When set relatively low in regards to the larger baud rates, the number of samples per bit is small. This can create errors and biases in either class of estimators. The sampling frequency should be set perhaps at least 20 times the largest expected band rate. Unfortunately, this means the weight vector must contain a large number of weights.

Another item which deserves attention is the weight vector. The weight vector is supposed to converge to a stable weight vector which will produce an error output of the ANC of the least power. In this sense the ADF filter weights will have converged to the optimal Wiener filter [1]. An analysis of the weight vector would be beneficial, especially in the case of a random bit input sequence as modeled in the simulation. Once the analysis is performed it will be possible to decide if the pseudo-correlation sub-class of estimators contains merit, and if they do, how well they will perform in comparison to the other proposed estimators.

Another aspect the analysis be concerned with is in the method of noise (or signal) cancelling. The method used here relies on the ANC and can be shown to be in the class of auto-regressive predictors [5]. This class of estimators is known for a spectrum which is characterized by peaks. The peaks result from a rational model for this estimator involving an all-pole transfer function. However, since the spectrum of the bit sequence is characterized more so by nulls in the frequency domain than by peaks, the bit sequence should be considered as a process other than an auto-regressive process. Therefore, it makes sense to use an estimator which is compatible with nulls, or in general, nulls and peaks. The auto-regressive, moving average estimator meets this requirement since its rational model is characterized with a transfer function containing both zeros and poles.

Appendix A: Computer Program Listings

<u>Program</u>	<u>page</u>
SBAUD	96
SADF	132
SSADF	145
BITLENGTH	151
ZCROSS	155
DLD	160

SBAUD

VAX-11 FORTRAN SOURCE FILENAME:

SBAUD.FOR

DEPARTMENT OF ELECTRICAL ENGINEERING

KANSAS STATE UNIVERSITY

REVISION

DATE

PROGRAMMER(S)

00.0

MAY 6, 1983

MARC D. BRACK

PURPOSE

This program is a simulation for a system that estimates the band rate of an FSK signal. The system consists of a delay-line discriminator (DLD) to demodulate the signal and of an adaptive noise canceller (ANC) to make the estimate. The results are compared to the results of the traditional method of zero crossing estimation.

ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE

DLD These routines are specialized routines written only for the purpose of the simulation. They are compatible with SBAUD and each other and are meant to be run in a single package.

SGOPEN These routines are general purpose routines that perform such functions as writing to disk, plotting, filtering, and making FFT's. Any other routines performing identical functions may be substituted for them.

ROUTINE(S) ACCESSED OR CALLED BY EMBEDDED ROUTINE(S)

BITLENGTH This is another specialized routine.

IMPORTANT: Weight vector length must be a power of two if the intermediate filter, the pseudo-correlation estimates, or the frequency response of the weight vector data are to be implemented. The DFT routine used in these applications requires a vector of length of a power of two.

Variables used in main routine:

ADF TRUE if ADF is to be included in simulation
 AI Amplitude of FSK signal envelope
 ARG Argument of FSK signal envelope
 AUTO TRUE if autocorrelation estimate is enabled
 BITS Number of bits in message
 BN Blank character for plotting purposes
 BOTH TRUE if two files are to be written to disk
 BUTER Complex butterworth filter routine
 BW Max bandwidth of signal according to sampling frequency
 C User response as Y or N
 CHOICE Pointer of directed GOTO statements
 C1 First estimate to be written
 C2 Number of estimates to be written
 DELAY Number of samples to delay in ANC
 DW FSK frequency shift in radians
 FC Center frequency of FSK signal
 FFT TRUE if DFT is to be made of weight vector
 FLAG Indicates sin- or cosine-type discriminator
 FO Fundamental frequency of entire message
 FREQP Max frequency of frequency response to be plotted
 FS Sampling frequency
 FSK TRUE if FSK signal information is to be obtained
 FW FSK frequency shift in Hertz
 HT Horizontal title
 HU Horizontal units

C	I	Loop counter variable
C		
C	II	Loop counter variable
C		
C	ITER	Simulation loop counter variable
C		
C	J	Loop counter variable
C		
C	LIM	TRUE if limiter is to be included in simulation
C		
C	LTI	Left title
C		
C	LUN	Left units
C		
C	MAXF	Max of DFT of weight mean or variance
C		
C	MU	Gradient convergence rate constant
C		
C	N	Number of samples in signal
C		
C	NN	Increment factor for plotting purposes
C		
C	ND	Delay for DLD
C		
C	NOISE	TRUE if noise is to be added in respective iteration
C		
C	NSBWM	Normalized bandwidth estimate mean
C		
C	NSBWS	Normalized bandwidth standard deviation
C		
C	OFILT	TRUE if intermediate filtering is to be done
C		
C	P	Factor for either shifting up or down in FSK signal
C		
C	PI	The constant pi
C		
C	PNUM	Number of points to plot
C		
C	POSTA	TRUE if post-filtering is to be done for ANC
C		
C	POSTBW	Post-filter bandwidth for ANC
C		
C	POSTBZ	Post-filter bandwidth for ZC analysis (if POSTA false)
C		
C	PRE	TRUE if pre-filtering is to be done
C		
C	PREBW	Pre-filter bandwidth
C		
C	R	Band rate
C		
C	RTI	Right title
C		

```

C      RUN      Right nnita
C
C      SANCZC   TRUE if ZC analysia is to be done in ANC
C
C      SBW      Bandwidth estimate
C
C      SBWM     Bandwidth estimate mean
C
C      SBWS     Bandwidth estimate mean square
C
C      SBWV     Bandwidth estimate variance
C
C      SCORR    TRUE if psendo-correlation analysia is to be done
C
C      SMM      TRUE if intermediate estimatea are to be made
C
C      SNR      TRUE if noise is to be added to aignal
C
C      SNRDB    Signal-to-noiae ration in dB
C
C      SPB      Samples per bit
C
C      STITLE   Snb-title for plot
C
C      TB       Bit interval
C
C      TD       Time delay for discriminator
C
C      TF       Time interval for FSK signal
C
C      THETA    Phaae offaet of FSK signal
C
C      TTITLE   Major title for plot
C
C      TRIALS   Number of iterations for simnlation loop
C
C      TS       Sampling interval
C
C      VAR      Variance for noiae
C
C      WBIAS    TRUE if biaa is to be ionclnded in weight vector
C
C      WC       Center frequency in radians per second
C
C      WINDOW   TRUE if window is to be nued in DFT sitnations
C
C      WNUM     Number of weighta in weight vector
C
C      WRIT     TRUE if data ia to be written to disk
C
C      WINUM    Length of zero padded weight vector in DFT sitnations
C

```

```

C      XSCALE  Scaling of abscissa values for plotting purposes
C
C      ZC      TRUE if traditional ZC analysis is to be done
C
C*****
C
C      Arrays used in the main routine:
C
C      AUTOC   Used in obtaining autocorrelation
C
C      CNOISE  Noise to be added to signal
C
C      DLDNN   Outpnt of DLD for noiseless inpnt
C
C      EROR    Error outpnt of secondary ANC
C
C      ERORI   Error outpnt of intermediate ANC
C
C      FILT    DLD outpnt intended of tradistional zc analysis
C
C      FSIG    FSK signal magnitnde
C
C      LEFT    Left array for plotting
C
C      MDLD    DLD outpnt, general case
C
C      NAUTO   Noise autocorrelation
C
C      PFSK    FSK signal phase
C
C      RIGHT   Right array for plotting
C
C      S       FSK envelope and general purpose complex array
C
C      SAUTO   Message autocorrelation
C
C      SEQ     Bit aeqnence for of message
C
C      SFEST   Secondary band rate estimatea
C
C      SFESTM  Mean of secondary band rate eatimates
C
C      SFESTS  Mean square of secondary band rate estimatea
C
C      SFESTV  Variance of secondary band rate eatimates
C
C      SIEST   Intermediate band rate eatimates
C
C      SIESTM  Mean of intermediate band rate estimates
C
C      SIESTS  Mean square of intermediate band rate estimatea
C

```

```

C      SIESTV  Variance of intermediate baud rate estimates
C
C      WMEAN   Weight vector mean
C
C      WMEANA  Average of all weight vector means
C
C      WVAR    Weight vector variance
C
C      WVARA   Average of all weight vecto variances
C
C      Y       ADF output of secondary ANC
C
C      YI      ADF output of intermediate ANC
C
C      ZEST    Traditional ZC baud rate estimate
C
C      ZESTM   Mean of traditional ZC baud rate estimate
C
C      ZESTS   Mean square of traditional ZC baud rate estimate
C
C      ZESTV   Variance of traditional ZC baud rate estimate
C
C*****
C

```

IMPLICIT NONE

```

REAL THETA,BW,AI,FSIG(2048),FILT(2048),MDLD(2048),MU,ARG
REAL TD,FC,TF,PREBW,POSTBW,PFSK(2048),FW,DW,R,FO,DLN(2048)
REAL VAR,SNRDB,SAUTO(2048),NAUTO(2048),SBWV
REAL WMEAN(0:256,200),WVAR(0:256,200),WMEANA(0:256),WVARA(0:256)
REAL XSCALE,LEFT(2048),RIGHT(2048),SBW,SBWS,SBWM,MAXF
REAL ZEST(10,200),SIEST(10,200),SFEST(10,200),POSTBZ,FREQP
REAL ZESTM(10),SIESTM(10),SFESTM(10),NSBWM,NSBWS
REAL ZESTS(10),SIESTS(10),SFESTS(10)
REAL ZESTV(10),SIESTV(10),SFESTV(10)
REAL EROR(2048),Y(2048),ERORI(2048),YI(2048)
REAL*8 PI,WC,FS,TS,TB
INTEGER SPB,P,I,J,SEQ(64),BITS,N,NN,FLAG,WNUM,DELAY
INTEGER ITER,ND,CHOICE,TRIALS,PNUM,C1,C2,II,WNUM
COMPLEX S(2048),BUTER,CNOISE(2048),AUTOC(2048)
CHARACTER*80 BN,TITLE,LT,RT,STITLE,HT,HU,LUN,RUN
CHARACTER C*1
LOGICAL NOISE,WBIAS,AUTO,ZC,ADF,FSK,POSTA,WINDOW,SANCZC
LOGICAL PRE,SNR,OFILT,LIM,SMM,BOTH,WRIT,FFT,SCORR

```

```

NN = 1
BN = ' '
PI = 3.141592653589793D0
N = 1024
FS = 9600
FO = FS/N
TS = 1./FS
BW = FS/2.

```

```

FC = 500*FS
WC = 2*PI*FC
BITS = 32
SPB = N/BITS
TB = TS*SPB
R = 1.0/TB
AI = 1.0
THETA = 0.0
NOISE = .FALSE.

```

```

C
C*****
C
C  PARAMETER INPUT
C
C  AUTOCORRELATION ESTIMATION ENABLE
C
C      CALL PAUTO(AUTO)
C
C  FSK SIGNAL INFORMATION ENABLE
C
C      CALL PFSKINFO(FSK)
C
C  SIMULATION ADAPTIVE FILTERING SUBROUTINE ENABLE
C
C      CALL PADF(ADF)
C
C  ZERO CROSSING ALGORITHM ENABLE
C
C      CALL PZCROSS(ZC)
C
C  INPUT SIGNAL PARAMETERS
C
C      TYPE *, ' '
C      TYPE *, 'INPUT SIGNAL PARAMETERS:'
C
C  SEQUENCE FORMATION
C
C      CALL PSEQUENCE(SEQ,BITS)
C
C  FSK PARAMETERS
C
C      CALL PFSKSHIFT(FW)
C      DW = 2.0*PI*FW
C
C  NOISE SELECTION
C
C      CALL PNOISE(SNRDB,SNR)
C
C  DELAY LINE DISCRIMINATOR PARAMETERS
C
C      TYPE *, ' '

```

```

      TYPE *, 'DLD RECEIVER PARAMETERS:'
C
C FILTER SELECTION
C
      CALL PPREFILTER(PREBW,PRE)
      CALL PPOSTFILTER(POSTBW,POSTA,ZC,POSTBZ)
C
C LIMITER ENABLE
C
      CALL PLIM(LIM)
C
C DLD SHIFT PARAMETER
C
      CALL PDELAY(ND)
      ND = INT(FC/(6*PREBW))
C
C ADAPTIVE FILTER PARAMETERS
C
      IF(ADF) THEN
      TYPE *, ' '
      TYPE *, 'ADAPTIVE FILTERING PARAMETERS:'
C
C ADF DELAY SELECTION
C
      CALL PADFDELAY(DELAY)
C
C NUMBER OF ADF WEIGHTS SELECTION
C
      CALL PWEIGHTS(WNUM)
C
C GRADIENT CONSTANT SELECTION
C
      CALL PCONSTANT(MU)
C
C WEIGHTING BIAS SELECTION
C
      CALL PBIAS(WBIAS)
      TYPE *, ' '
      TYPE *, 'ADF ESTIMATOR SELECTION:'
C
C INTERMEDIATE FILTERING ENABLE
C
      CALL PFILT1(OFILT)
C
C MAX/MIN ESTIMATE ENABLE
C
      CALL PMM(SMM)
C
C PSEUDO-CORRELATION ESTIMATE ENABLE
C
      CALL PCORR(SCORR)

```

```

C
C  ANC-ZC ESTIAMTE ENABLE
C
C      CALL PANCZC(SANCZC)
C
C      ENDIF
C
C  NUMBER OF TRIALS SELECTION
C
C      CALL PTRIALS(TRIALS)
C
C*****
C
C  RUN INITIALIZATION
C
1000  CONTINUE

      SBWM = 0
      SBWS = 0
      NSBWM = 0
      NSBWS = 0
      DO 140 I=1,10
          SIESTM(I) = 0
          SIESTS(I) = 0
          ZESTM(I) = 0
          ZESTS(I) = 0
          SFESTM(I) = 0
          SFESTS(I) = 0
140  CONTINUE

      NOISE = .FALSE.

      DO 150 J=0,WNUM
          WMEANA(J) = 0.
          WVARA(J) = 0.
150  CONTINUE

      IF (AUTO) THEN
          DO 160 I=1,N
160      NAUTO(I) = 0.
      ENDIF
C
C*****
C
C  SIMULATION DO LOOP
C
      DO 1500 ITER=1,TRIALS

          TYPE *, ' '
          TYPE *, 'IN ITERATION NUMBER ',ITER

1100  CONTINUE

```

```

C
C*****
C
C FSK GENERATION
C
      DO 180 I=1,BITS
          IF (SEQ(I).EQ.1) THEN
              P = 1
          ELSE
              P = -1
          ENDIF
C
C MODULATING FUNCTION
C
      DO 170 J=1,SPB
          S((I-1)*SPB+J) = AI*CDEXP(DCMPLX(0.0D0,P*((I-1)*SPB+J-1)*TS*DW))
170      CONTINUE
180      CONTINUE
C
C*****
C
C NOISE ADDITION
C
      IF (NOISE) THEN
          CALL SNOISE(CNOISE,N,AI,TS,PBWB,SNRDB,SNR)
          DO 190 I=1,N
              S(I) = S(I) + CNOISE(I)
190      ENDIF
C
C OBTAIN FREQUENCY COMPONENTS OF FSK
C
      IF(FSK) THEN
          CALL DFT(S,N,0)
          DO 200 I=1,N
              FSIG(I) = CABS(S(I))
200      CONTINUE
          CALL DFT(S,N,1)
C
C OBTAIN PHASE OF FSK ENVELOPE
C
          DO 210 I=1,N
              PFSK(I) = ATAN2(AIMAG(S(I)),REAL(S(I)))/PI
210      ENDIF
C
C*****
C
C DELAY-LINE DISCRIMINATOR
C

```

```

C
C   PRE-FILTERING
C
      IF (PRE) THEN
        CALL DFT(S,N,0)
        DO 300 I=1,N/2
          S(I) = S(I)*BUTER(4,PREBW,FLOAT(I-1)*FO)
          S(N+1-I) = S(N+1-I)*BUTER(4,PREBW,FLOAT(-I)*FO)
300      CONTINUE
          S(N/2+1) = CMPLX(0.,0.)
          CALL DFT(S,N,1)
        ENDIF
C
C   LIMITER
C
      IF (LIM) THEN
        DO 305 I=1,N
          IF(.NOT.(REAL(S(I)).EQ.0 .AND. AIMAG(S(I)).EQ.0))THEN
            ARG = ATAN2(AIMAG(S(I)),REAL(S(I)))
            ELSE
              ARG = 0
            ENDIF
            S(I) = AI*CMPLX(COS(ARG),SIN(ARG))
305      CONTINUE
        ENDIF
C
C   DLD
C
      TD = ND/(2.0*FC)
      FLAG = 1
      TF = N*TS

      CALL DLD(TD,FC,FLAG,S,N,TF)
C
C   POST-FILTERING FOR ADAPTIVE FILTERING
C
      IF (POSTA) THEN
        CALL DFT(S,N,0)
        DO 310 I=1,N/2
          S(I) = S(I)*BUTER(4,POSTBW,FLOAT(I-1)*FO)
          S(N+1-I) = S(N+1-I)*BUTER(4,POSTBW,FLOAT(-I)*FO)
310      CONTINUE
          S(N/2+1) = CMPLX(0.,0.)
          CALL DFT(S,N,1)
        ENDIF
C
C*****
C
C   INITIAL NOISELESS CASE
C
      IF (.NOT.NOISE.AND.SNR) THEN

```

```

      NOISE = .TRUE.
      DO 400 I=1,N
400      DLDNN(I) = REAL(S(I))
C
C  MESSAGE AUTOCORREALATION
C
      IF (AUTO) THEN
        DO 460 I=1,N
460      AUTOC(I) = CMPLX(DLDNN(I),0.)
        CALL DFT(AUTOC,N,0)

        DO 470 I=1,N
470      AUTOC(I) = CMPLX(CABS(AUTOC(I))**2,0.)
        CALL DFT(AUTOC,N,1)

        DO 480 I=1,N
480      SAUTO(I) = CABS(AUTOC(I))
      ENDIF

      GOTO 1100
    ENDIF

C
C*****
C
C  NOISE AUTOCORRELATION ESTIMATE
C
      IF (AUTO) THEN
        DO 410 I=1,N
410      AUTOC(I) = CMPLX(REAL(S(I))-DLDNN(I),0.)
        CALL DFT(AUTOC,N,0)

        DO 420 I=1,N
420      AUTOC(I) = CMPLX(CABS(AUTOC(I))**2,0.)
        CALL DFT(AUTOC,N,1)

        DO 430 I=1,N
430      NAUTO(I) = NAUTO(I) + CABS(AUTOC(I))/TRIALS
      ENDIF

      DO 440 I=1,N
440      MDLD(I) = REAL(S(I))
C
C  PERFORM ZERO CROSSING ESTIMATE, IF NO ADF POST FILTER, INCLUDE FILTERING
C
      IF(ZC) THEN
        IF(.NOT.POSTA) THEN
          DO 441 I=1,N
441      CNOISE(I) = S(I)
          CALL DFT(CNOISE,N,0)
          DO 442 I=1,N/2
            CNOISE(I) = CNOISE(I)*BUTER(4,POSTBZ,FLOAT(I-1)*FO)
            CNOISE(N+1-I) = CNOISE(N+1-I)*BUTER(4,POSTBZ,FLOAT(-I)*FO)

```

```

442      CONTINUE
          CNOISE(N/2+1) = CMPLX(0.,0.)
          CALL DFT(CNOISE,N,1)

          DO 443 I=1,N
443      FILT(I) = REAL(CNOISE(I))

          CALL ZCROSS(FILT,N,ZEST(1,ITER),1)
          ELSE
          CALL ZCROSS(MDLN,N,ZEST(1,ITER),1)
          ENDIF
        ENDIF
C
C*****
C
C ADAPTIVE NOISE FILTERING SUBROUTINE
C
      IF (ADF) THEN

          CALL SSADF(MDLN,N,MU,WNUM,DELAY,WBIAS,WMEAN(0,ITER),WVAR(0,ITER),
X          OFILT,SFEST(1,ITER),TS,SIEST(1,ITER),SBW,SMM,ERORI,YI,
X          EROR,Y,SCORR,SANCZC)
C
C BANDWIDTH ESTIMATE
C
      IF(OFILT) THEN
          SBWM = SBWM + SBW/TRIALS
          SBWS = SBWS + SBW**2/TRIALS
          NSBWM = NSBWM + ((SBW - R)/R)/TRIALS
          NSBWS = NSBWS + ((SBW - R)/R)**2/TRIALS
      ENDIF
C
C BAUD RATE ESTIMATE STATISTICAL UPDATE
C
      IF(SMM) THEN
          DO 444 I=1,10
              SIESTM(I) = SIESTM(I) + ((SIEST(I,ITER)-SPB)/SPB)/TRIALS
              SIESTS(I) = SIESTS(I) + ((SIEST(I,ITER)-SPB)/SPB)**2/TRIALS
444      CONTINUE
          ENDIF
          IF (OFILT) THEN
              DO 445 I=1,10
                  SFESTM(I) = SFESTM(I) + ((SFEST(I,ITER)-SPB)/SPB)/TRIALS
                  SFESTS(I) = SFESTS(I) + ((SFEST(I,ITER)-SPB)/SPB)**2/TRIALS
445      CONTINUE
          ENDIF

          DO 450 I=0,WNUM
              WMEANA(I) = WMEANA(I) + WMEAN(I,ITER)/TRIALS
              WVARA(I) = WVARA(I) + WVAR(I,ITER)/TRIALS
450      CONTINUE

```

```

ENDIF

IF(ZC) THEN
  DO 455 I=1,1
    ZESTM(I) = ZESTM(I) + ((ZEST(I,ITER)-SPB)/SPB)/TRIALS
    ZESTS(I) = ZESTS(I) + ((ZEST(I,ITER)-SPB)/SPB)**2/TRIALS
455    CONTINUE
  ENDF

1500  CONTINUE
C
C  END OF SIMULATION LOOP
C
C*****
C
C  OBTAIN VARIANCE OF STATISTICS
C
  IF (ADF) THEN
    IF(OFILT) THEN
      SBWV = SBWS - SBWM**2
      IF(SBWV.GE.0.) SBWV = SBWV**0.5
      NSBWS = NSBWS**0.5
    ENDF
    IF (SMM) THEN
      DO 490 I=1,10
        SIESTV(I) = SIESTS(I) - SIESTM(I)**2
        IF(SIESTV(I).GE.0.0) SIESTV(I) = SIESTV(I)**0.5
490      CONTINUE
    ENDF
    IF(OFILT) THEN
      DO 495 I=1,10
        SFESTV(I) = SFESTS(I) - SFESTM(I)**2
        IF(SFESTV(I).GE.0.0) SFESTV(I) = SFESTV(I)**0.5
495      CONTINUE
    ENDF
  ENDF

  IF (ZC) THEN
    DO 500 I=1,1
      ZESTV(I) = ZESTS(I) - ZESTM(I)**2
      IF(ZESTV(I).GE.0.0) ZESTV(I) = ZESTV(I)**0.5
500    CONTINUE
  ENDF
C
C*****
C
C  PLOTTING AND MODIFICATION
C
  CALL PTITLE(TITLE,PRE,POSTA,FW,PREBW,POSTBW,SNRDB,SNR,R,LIM)
  IF (WBIAS) THEN
    WRITE(STITLE,14) 'ADF-DELAY= ',DELAY,', N=',WNUM,', MU=',MU,', BIAS'
  ELSE

```

```
WRITE(STITLE,14) 'ADF-DELAY= ',DELAY,', N=',WNUM,', MU=',MU
ENDIF
```

```
2000  TYPE *, ' '
      TYPE 22
22    FORMAT(' Do you want to:',/
x      , ' 1. Plot',/
x      , ' 2. Change parameters',/
x      , ' 3. Enter user controlled ADF routine',/
x      , ' 4. Write results',/
x      , ' 5. Quit',/
x      , ' 6. Write output to disk')
      TYPE 11, '/Enter selection: '
      READ (*,*) CHOICE
```

```
WRIT = .FALSE.
IF(CHOICE.EQ.6) WRIT = .TRUE.
```

```
GOTO (5000,6000,7000,8000,9000,5000) CHOICE
```

```
C
C*****
C
```

```
C PLOTTING AND WRITING TO DISK
```

```
C
5000  TYPE *, ' '
      TYPE 23
23    FORMAT(' Do you want to plot(write):',/
x      , ' 1. DLD output',/
x      , ' 2. Frequency components of FSK signal',/
x      , ' 3. Phase of FSK envelope',/
x      , ' 4. Autocorrelation estimates',/
x      , ' 5. Weight mean and variance data',/
x      , ' 6. Intermediate ANC error',/
x      , ' 7. Intermediate ADF output',/
x      , ' 8. Secondary ANC input',/
x      , ' 9. Secondary ANC error',/
x      , ' 10. Secondary ADF output',/
x      , ' 11. Or do you want to return to main menu')
      TYPE 11, '/Enter selection: '
      READ(*,*) CHOICE
```

```
GOTO(5100,5200,5300,5400,5500,5600,5700,5750,5800,5900,2000) CHOICE
```

```
C
C*****
C
```

```
C DLD OUTPUT
```

```
C
5100  DO 5110 I=1,N
5110  LEFT(I) = REAL(S(I))
```

```
WINUM = N
BOTH = .FALSE.
```

```
IF(WRIT) GOTO 5998
```

```
LTI = 'REAL PART OF DLD OUTPUT'
LUN = ' '
RTI = 'NONE'
HT = 'SAMPLE NUMBER'
HU = ' '
STITLE = ' '
XSCALE = 1
PNUM = N
```

```
GOTO 5999
```

```
C
```

```
C*****
```

```
C
```

```
C FREQUENCY COMPONENTS OF FSK SIGNAL
```

```
C
```

```
5200 IF(.NOT.FSK) THEN
```

```
TYPE *, ' '
```

```
TYPE *, 'NOT ENABLED'
```

```
GOTO 2000
```

```
ENDIF
```

```
DO 5210 I=1,N
```

```
5210 LEFT(I) = FSIQ(I)
```

```
WINUM = N
```

```
BOTH = .FALSE.
```

```
IF(WRIT) GOTO 5998
```

```
LTI = 'FREQUENCY COMPONENTS BEFORE DLD'
```

```
LUN = ' '
```

```
RTI = 'NONE'
```

```
HT = 'FREQUENCY'
```

```
HU = 'HERTZ'
```

```
STITLE = ' '
```

```
XSCALE = FO
```

```
PNUM = N/2
```

```
GOTO 5999
```

```
C
```

```
C*****
```

```
C
```

```
C PHASE OF FSK ENVELOPE
```

```
C
```

```
5300 IF(.NOT.FSK) THEN
```

```
TYPE *, ' '
```

```
TYPE *, 'NOT ENABLED'
```

```
GOTO 2000
```

```
ENDIF
```

```
DO 5310 I=1,N
```

```
5310 LEFT(I) = PFSK(I)
```

```

WINUM = N
BOTH = .FALSE.
IF(WRIT) GOTO 5998

LTI = 'PHASE OF FSK ENVELOPE'
LUN = 'PI RADIANS'
RTI = 'NONE'
HT = 'SAMPLE NUMBER'
HU = ' '
STITLE = ' '
XSCALE = 1
PNUM = N

```

```

GOTO 5999

```

```

C
C*****
C
C  AUTOCORRELATION ESTIMATES
C
5400  IF(.NOT.AUTO) THEN
      TYPE *, ' '
      TYPE *, ' AUTOCORRELATION ESTIMATE NOT ENABLED'
      GOTO 2000
    ENDIF

    TYPE *, ' '
    TYPE 9, 'Plot how many points? (max=',N,') '
    READ (*,*) PNUM

    DO 5410 I=1,PNUM
      LEFT(I) = SAUTO(I)
      RIGHT(I) = NAUTO(I)
5410  CONTINUE

    WINUM = PNUM
    BOTH = .TRUE.
    IF(WRIT) GOTO 5998

    LTI = 'SIGNAL AUTOCORRELATION'
    LUN = ' '
    RTI = 'NOISE AUTOCORRELATION'
    RUN = ' '
    HT = 'LAG COMPONENT'
    HU = ' '
    STITLE = ' '
    XSCALE = 1

    GOTO 5999

C
C*****
C
C  WEIGHT MEAN AND VARIANCE DATA

```

```

C
C PLOT OR WRITE DATA OR DFT OF DATA
C
5500  TYPE *, ' '
      TYPE 11, '/Do you want weight sequence or DFT of the sequence? (W/D) '
      READ (*,12) C
      FFT = .FALSE.
      IF(C.EQ.'D' .OR. C.EQ.'d') THEN
        FFT = .TRUE.
        TYPE 11, '/Do you want to window the sequence? (Y/N) '
        READ(*,12) C
        IF(C.EQ.'Y' .OR. C.EQ.'y') THEN
          WINDOW = .TRUE.
        ELSE
          WINDOW = .FALSE.
        ENDIF
        IF(.NOT.WRIT)THEN
          TYPE *, ' '
          TYPE 11, '/Plot up to what frequency? '
          READ(*,*) FREQP
        ENDIF
      ENDIF
      TYPE *, ' '

C
C PLOT OR WRITE AVERAGE DATA OR DATA OF ONE TRIAL
C
      TYPE 15
15    FORMAT (' Plot (write) either: ',/
x      , ' 0. Average of mean and variance',/
x      , ' #. Mean and variance of one run')
      TYPE 16 , '/Enter selection as either 0 or run number (up to ',
x      TRIALS,'): '
      READ(*,*) CHOICE

      IF(CHOICE.EQ.0) THEN
        IF(FFT) THEN
          TYPE*, ' '
          TYPE*, 'DFT NOT VALID FOR AVERAGE'
          GOTO 5500
        ENDIF
        DO 5510 J=0,WNUM
          LEFT(J+1) = WMEANA(J)
          RIGHT(J+1) = WVARA(J)
5510    CONTINUE

        WINUM = WNUM + 1
        BOTH = .TRUE.
        IF(WRIT) GOTO 5998

C
C AVERAGE DATA
C
      LTI = 'AVERAGED MEAN'

```

```

LUN = ' '
RTI = 'AVERAGED VARIANCE'
RUN = ' '
WRITE(HT,13) 'WEIGHT MEAN AND VARIANCE AVERAGE, ', TRIALS, ' TRIALS'
HU = ' '
PNUM = WNUM + 1
XSCALE = 1

GOTO 5999

ELSEIF(CHOICE.GT.0 .AND. CHOICE.LE.TRIALS) THEN
C
C DATA OF INDIVIDUAL TRIAL
C
    IF(.NOT.FFT) THEN
        DO 5520 J=0,WNUM
            LEFT(J+1) = WMEAN(J,CHOICE)
            RIGHT(J+1) = WVAR(J,CHOICE)
5520        CONTINUE

            WINUM = WNUM+1
            BOTH = .TRUE.
            IF(WRIT) GOTO 5998

            LTI = 'WEIGHT MEAN'
            LUN = ' '
            RTI = 'WEIGHT VARIANCE'
            RUN = ' '
            HT = 'MEAN AND VARIANCE OF WEIGHT VECTORS'
            HU = ' '
            PNUM = WNUM + 1
            XSCALE = 1
            GOTO 5999
        ELSE
C
C DFT OF DATA
C
            DO 5530 J=0,WNUM
                AUTOC(J+1) = CMPLX(WMEAN(WNUM-J,CHOICE),0.)
                IF(WINDOW) AUTOC(J+1) = AUTOC(J+1) * FLOAT(J)/WNUM
5530            CONTINUE
                DO 5535 J=WNUM+1,2*WNUM
                    AUTOC(J+1) = CMPLX(WMEAN(J-WNUM,CHOICE),0.)
                    IF(WINDOW) AUTOC(J+1) = AUTOC(J+1) * (1.0 - FLOAT(J-WNUM)/WNUM)
5535                CONTINUE
                DO 5540 J=2*WNUM+1,8*WNUM-1
                    AUTOC(J+1) = CMPLX(0.,0.)
5540                CALL DFT(AUTOC,8*WNUM,0)
                    MAXF = 0.
                    DO 5550 J=1,4*WNUM
5550                        MAXF = MAX(CABS(AUTOC(J)),MAXF)
                    DO 5560 J=1,4*WNUM

```

```

5560      LEFT(J) = 20*ALOG10(CABS(AUTOC(J))/MAXF)

      DO 5570 J=0,WNUM
        AUTOC(J+1) = CMPLX(WVAR(WNUM-J,CHOICE),0.)
        IF(WINDOW) AUTOC(J+1) = AUTOC(J+1) * FLOAT(J)/WNUM
5570      CONTINUE
      DO 5575 J=WNUM+1,2*WNUM
        AUTOC(J+1) = CMPLX(WVAR(J-WNUM,CHOICE),0.)
        IF(WINDOW) AUTOC(J+1) = AUTOC(J+1) * (1.0 - FLOAT(J-WNUM)/WNUM)
5575      CONTINUE
      DO 5580 J=2*WNUM+1,8*WNUM-1
5580      AUTOC(J+1) = CMPLX(0.,0.)
      CALL DFT(AUTOC,8*WNUM,0)
      MAXF = 0.
      DO 5590 J=1,4*WNUM
5590      MAXF = MAX(CABS(AUTOC(J)),MAXF)
      DO 5595 J=1,4*WNUM
5595      RIGHT(J) = 20*ALOG10(CABS(AUTOC(J))/MAXF)

      WINUM = WNUM*4
      BOTH = .TRUE.
      IF(WRIT) GOTO 5998

      LTI = 'DFT OF WEIGHT VECTOR MEAN'
      LUN = ' '
      RTI = 'DFT OF WEIGHT VECTOR VARIANCE'
      RUN = ' '
      HT = 'FREQUENCY'
      HU = 'HERTZ'
      XSCALE = FS/(WNUM*8)
      PNUM = FREQP/XSCALE
      GOTO 5999

      ENDIF
    ELSE

      TYPE *, ' '
      TYPE *, ' CHOICE NOT WITHIN VALID RANGE, TRY AGAIN'

      GOTO 5500

    ENDIF

  C
  C*****
  C
  C INTERMEDIATE ANC ERROR
  C
5600  IF(.NOT.OFIL) THEN
      TYPE*, 'INTERMEDIATE FILTERING NOT ENABLED'
      GOTO 5000
    ENDIF

```

```

DO 5610 I=1,N
5610     LEFT(I) = ERORI(I)

WINUM = N
BOTH = .FALSE.
IF(WRIT) GOTO 5998

LTI = 'INTERMEDIATE ANC ERROR'
LUN = ' '
RTI = 'NONE'
HT = 'SAMPLE NUMBER'
HU = ' '
STITLE = ' '
XSCALE = 1
PNUM = N

GOTO 5999

C
C*****
C
C  INTERMEDIATE ADF OUTPUT
C
5700     IF(.NOT.OFILT) THEN
           TYPE*, 'INTERMEDIATE FILTERING NOT ENABLED'
           GOTO 5000
        ENDIF

DO 5710 I=1,N
5710     LEFT(I) = YI(I)

WINUM = N
BOTH = .FALSE.
IF(WRIT) GOTO 5998

LTI = 'INTERMEDIATE ADF OUTPUT'
LUN = ' '
RTI = 'NONE'
HT = 'SAMPLE NUMBER'
HU = ' '
STITLE = ' '
XSCALE = 1
PNUM = N

GOTO 5999

C
C*****
C
C  SECONDARY ANC INPUT
C
5750     IF(.NOT.OFILT) THEN
           TYPE*, 'INTERMEDIATE FILTERING NOT ENABLED'
           GOTO 5000

```

ENDIF

DO 5760 I=1,N
5760 LEFT(I) = MDLD(I)

WINUM = N
BOTH = .FALSE.
IF(WRIT) GOTO 5998

LTI = 'INTERMEDIATE ANC INPUT'
LUN = ' '
RTI = 'NONE'
HT = 'SAMPLE NUMBER'
HU = ' '
STITLE = ' '
XSCALE = 1
PNUM = N

GOTO 5999

C
C*****

C
C SECONDARY ANC ERROR
C

5800 DO 5810 I=1,N
5810 LEFT(I) = EROR(I)

WINUM = N
BOTH = .FALSE.
IF(WRIT) GOTO 5998

LTI = 'ANC ERROR'
LUN = ' '
RTI = 'NONE'
HT = 'SAMPLE NUMBER'
HU = ' '
STITLE = ' '
XSCALE = 1
PNUM = N

GOTO 5999

C
C*****

C
C SECONDARY ADF OUTPUT
C

5900 DO 5910 I=1,N
5910 LEFT(I) = Y(I)

WINUM = N
BOTH = .FALSE.
IF(WRIT) GOTO 5998

```

LTI = 'ADF OUTPUT'
LUN = ' '
RTI = 'NONE'
HT = 'SAMPLE NUMBER'
HU = ' '
STITLE = ' '
XSCALE = 1
PNUM = N

```

GOTO 5999

```

5998      TYPE *, ' '
          CALL SGOPEN(10,'WRITE','Filenama? ', 'NONAME', 'REAL', WINUM)
          CALL SGTRAN(10,'WRITE', 'REAL', LEFT, WINUM)

          IF(BOTH) THEN
              CALL SGOPEN(10,'WRITE','Filenama? ', 'NONAME', 'REAL', WINUM)
              CALL SGTRAN(10,'WRITE', 'REAL', RIGHT, WINUM)
          ENDIF
          GOTO 2000

5999      CALL SPLOT(LEFT, RIGHT, PNUM, NN, LTI, LUN, RTI, RUN, HT, HU, TITLE,
X          STITLE, XSCALE)
          READ(*,12) C
          GOTO 2000

```

```

C*****
C
C  MODIFY PARAMETERS
C
6000  TYPE *, ' '
      TYPE 17
17    FORMAT(' Do you want to change:',/
x      , '      1. Bit aacquanca',/
x      , '      2. FSK frequency shift',/
x      , '      3. Signal to noise ratio',/
x      , '      4. Prefilter bandwidth',/
x      , '      5. Limiter enable',/
x      , '      6. Postfilter bandwidth',/
x      , '      7. ADF delay',/
x      , '      8. Number of weights',/
x      , '      9. Gradient constant',/
x      , '     10. ADF weight bias enable',/
x      , '     11. ANC pseudo-correlation estimate enabling',/
x      , '     12. ANC max/min estimate enabling',/
x      , '     13. ANC zero crossing estimate enabling',/
x      , '     14. ANC output filter enabling',/
x      , '     15. Autocorrelation enabling',/
x      , '     16. FSK signal information enabling',/

```

```

x      , '      17. Simulation ADF enabling', /
x      , '      18. Zero crossing enabling', /
x      , '      19. Number of trials per run', /
x      , '      20. Or do you want to make a run', /
x      , '      21. Or do you want to return to the main menu')
TYPE 11, 'Enter selection: '
READ(*,*) CHOICE

      GOTO (6010,6020,6030,6040,6050,6060,6070,6080,6090,6100,6110,
X      6120,6130,6140,6150,6160,6170,6180,6190,1000,2000) CHOICE

6010   CALL PSEQUENCE(SEQ,BITS)
      GOTO 6000

6020   CALL PFSKSHIFT(FW)
      GOTO 6000

6030   CALL PNOISE(SNRDB,SNR)
      GOTO 6000

6040   CALL PPREFILTER(PREBW,PRE)
      GOTO 6000

6050   CALL PLIM(LIM)
      GOTO 6000

6060   CALL PPOSTFILTER(POSTBW,POSTA,ZC,POSTBZ)
      GOTO 6000

6070   CALL PADFDELAY(DELAY)
      GOTO 6000

6080   CALL PWEIGHTS(WNUM)
      GOTO 6000

6090   CALL PCONSTANT(MU)
      GOTO 6000

6100   CALL PBIAS(WBIAS)
      GOTO 6000

6110   CALL PCORR(SCORR)
      GOTO 6000

6120   CALL PMM(SMM)
      GOTO 6000

6130   CALL PANCZC(SANCZC)
      GOTO 6000

6140   CALL PFILT1(OFILT)
      GOTO 6000

```

```

6150  CALL PAUTO(AUTO)
      GOTO 6000

6160  CALL PFSKINFO(FSK)
      GOTO 6000

6170  CALL PADF(ADF)
      GOTO 6000

6180  CALL PZCROSS(ZC)
      GOTO 6000

6190  CALL PTRIALS(TRIALS)
      GOTO 6000

C
C*****
C
C  ENTER USER CONTROLLED ADAPTIVE FILTERING ROUTINE
C
7000  DO 7100 I=1,N
      MDLD(I) = REAL(S(I))
7100  CONTINUE
      CALL SADF(MDLD,DLDNN,N,TS,TITLE)
      GOTO 2000

C
C*****
C
C  WRITE PARAMETERS, STATISTICS, AND ESTIMATES
C
8000  TYPE *,' '
      TYPE 24
24    FORMAT (' Do you want to write:',/,
x      '      1. Simulation parameters',/,
x      '      2. Estimate error statistics',/,
x      '      3. Band rate estimates',/,
x      '      4. Everything')
      TYPE 11, '/Enter selection: '
      READ(*,*) CHOICE

C
C  NUMBER OF ESTIMATES TO WRITE
C
      IF(CHOICE.EQ.3 .OR. CHOICE.EQ.4) THEN
        TYPE *,' '
        TYPE 25, ' There are ',TRIALS,' estimates'
        TYPE 11, '/Enter first estimate and number of estimates: '
        READ(*,*) C1,C2
        ENDIF

        GOTO (8100,8200,8300,8100) CHOICE

C
C*****

```

```

C
C  PARAMETERS
C
8100  WRITE(*,26) TRIALS,SNRDB,N,FS,BITS,R,FC,FW,2*PREBW
26    FORMAT(' ',/,/,
X      ' SIMULATION PARAMETERS:',/,
X      /, ' Simulation Parameters:',/,
X      ' Number of trials per run = ',I3,/,
X      ' Signal-to-noise ratio = ',F4.1,' dB',/,
X      /, ' Sampling Parameters:',/,
X      ' Number of samples = ',I4,/,
X      ' Sampling Frequency = ',F6.0,' Hz',/,
X      /, ' Bit Sequence Parameters:',/,
X      ' Number of bits = ',I2,/,
X      ' Baud rate = ',F5.0,' Hz',/,
X      /, ' FSK Modulation Parameters:',/,
X      ' Carrier frequency = ',F8.0,' Hz',/,
X      ' Frequency shift = ',F5.0,' Hz',/,
X      /, ' Receiver Parameters:',/,
X      ' Pre-filter bandwidth = ',F5.0,' Hz')
      IF(.NOT.POSTA) THEN
        WRITE(*,*) ' Post filtering not enabled for ANC'
        IF(ZC) WRITE(*,31) POSTBZ
31      FORMAT(' Post-filter bandwidth for ZC = ',F5.0,' Hz')
      ELSE
        WRITE(*,32) POSTBW
32      FORMAT(' Post-filter bandwidth = ',F5.0,' Hz')
      ENDIF
      IF(LIM) THEN
        WRITE(*,*) ' Limiter enabled'
      ELSE
        WRITE(*,*) ' Limiter not enabled'
      ENDIF
      WRITE(*,27) DELAY,MU,WNUM
27    FORMAT(' ',/,
X      ' Adaptive Noise Cancelling Parameters:',/,
X      ' Reference input delay = ',I2,/,
X      ' Gradient constant = ',E8.1,/,
X      ' Number of weights = ',I3)

      IF(WBIAS) THEN
        WRITE(*,*) ' Weighting bias included'
      ELSE
        WRITE(*,*) ' Weighting bias not included'
      ENDIF

      WRITE(*,*) ' '
      WRITE(*,*) ' '
      IF (ADF .OR. ZC) WRITE (*,*) 'ESTIMATORS ENABLED:'
      WRITE(*,*) ' '
      IF(OFILT) WRITE(*,*) ' ANC Intermediate Filtering enabled'
      IF(SMM) WRITE(*,*) ' Min/Max estimators enabled'

```

```

IF(SANCZC) WRITE(*,*) '    ANC-ZC estimators enabled'
IF(SCORR) WRITE(*,*) '    Pseudo-Correlation estimators enabled'
IF(ZC) WRITE(*,*) '    Zero Crossing estimator enabled'

IF(CHOICE.EQ.1) GOTO 2000

C
C*****
C
C WRITE STATISTICS
C
      WRITE(*,28)
8200  WRITE(*,*) ' '
      WRITE(*,*) ' '
      WRITE(*,*) 'NORMALIZED BAUD RATE ESTIMATE ERROR STATISTICS:'
      WRITE(*,*) ' '

C
C INTERMEDIATE ESTIMATE STATISTICS
C
      IF (ADF) THEN
        WRITE(*,*) ' '
        WRITE(*,*) ' MMI    MM2I    AMI    MLTOI    MZCI    MAI    VAI    ',
X        ' EXCI'
        WRITE(*,*) ' '
        WRITE(*,*) 'Normalized baud rate mean error:'
        WRITE(*,19) (SIESTM(I),I=1,8)
        WRITE(*,*) 'Normalized baud rate error variance:'
        WRITE(*,19) (SIESTV(I),I=1,8)
        WRITE(*,*) 'Normalized baud rate mean squared error:'
        WRITE(*,19) (SIESTS(I),I=1,8)
      ENDIF

C
C SECONDARY ESTIMATE STATISTICS
C
      IF (OFILT) THEN
        WRITE(*,*) ' '
        WRITE(*,*) ' MMF    MM2F    AMF    MLTOF    MZCF    MAF    VAF    ',
X        ' XZCF    YZCF    EZCF'
        WRITE(*,*) ' '
        WRITE(*,*) 'Normalized baud rate mean error:'
        WRITE(*,19) (SFESTM(I),I=1,10)
        WRITE(*,*) 'Normalized baud rate error variance:'
        WRITE(*,19) (SFESTV(I),I=1,10)
        WRITE(*,*) 'Normalized baud rate mean squared error:'
        WRITE(*,19) (SFESTS(I),I=1,10)

C
C BANDWIDTH STATISTICS
C
      IF (OFILT) THEN
        WRITE(*,*) ' '
        WRITE(*,*) 'BANDWIDTH ESTIMATE:'
        WRITE(*,*) ' '
        WRITE(*,30) ' Bandwidth mean = ',SBWM,

```

```

X          '      Bandwidth standard deviation = ',SBWV
      WRITE(*,33) ' Normalized bandwidth mean = ',NSBWM,
X          '      Standard deviation = ',NSBWS
      ENDIF
ENDIF
C
C TRADITIONAL ZERO CROSSING STATISTICS
C
      IF (ZC) THEN
        WRITE(*,*) ' '
        WRITE(*,*) 'ZERO CROSSING ESTIMATE ERROR STATISTICS:'
        WRITE(*,*) 'Normalized baud rate mean error:'
        WRITE(*,19) (ZESTM(I),I=1,1)
        WRITE(*,*) 'Normalized baud rate error variance:'
        WRITE(*,19) (ZESTV(I),I=1,1)
        WRITE(*,*) 'Normalized baud rate mean squared error:'
        WRITE(*,19) (ZESTS(I),I=1,1)
      ENDIF
      WRITE(*,*) ' '

      IF(CHOICE.EQ.2) GOTO 2000
C
C*****
C
C WRITE ESTIMATES
C
      WRITE(*,28)
8300      WRITE(*,*) ' '
      WRITE(*,*) ' '
      WRITE(*,*) 'BIT LENGTH ESTIMATES (IN NUMBER OF SAMPLES):'
      WRITE(*,*) ' '

C
C INTERMEDIATE ESTIMATES
C
      IF (ADF) THEN
        WRITE(*,*) ' '
        WRITE(*,*) ' MMI    MM2I    AMI    MLTOI    MZCI    MAI    VAI    ',
X          ' EZCIT'
        WRITE(*,*) ' '
        DO 8310 II=C1,C1+C2-1
          WRITE(*,29) (SIEST(I,II),I=1,8)
          IF(MOD(II,10).EQ.0) WRITE(*,*) ' '
8310      CONTINUE
      ENDIF
C
C SECONDARY ESTIMATES
C
      IF(OFILT) THEN
        WRITE(*,*) ' '
        WRITE(*,*) ' MMF    MM2F    AMF    MLTOF    MZCF    MAF    VAF    ',
X          ' XZCF    YZCF    EZCF'
        WRITE(*,*) ' '

```

```

      DO 8340 II=C1,C1+C2-1
        WRITE(*,29) (SFEST(I,II),I=1,10)
        IF(MOD(II,10).EQ.0) WRITE(*,*) ' '
8340    CONTINUE
      ENDIF

C
C  TRADISTION ZERO CROSSING ESTIMATES
C
      IF(ZC) THEN
        WRITE(*,*) ' '
        WRITE(*,*) 'Zero Crossing Estimates:'
        WRITE(*,*) ' '
        DO 8360 II=C1,C1+C2-1
          WRITE(*,29) (ZEST(I,II),I=1,1)
          IF(MOD(II,10).EQ.0) WRITE(*,*) ' '
8360    CONTINUE
        WRITE(*,*) ' '
      ENDIF

      GOTO 2000

C
C*****
C
C  FORMAT STATEMENTS AND MAJOR PROGRAM ENDING
C
9000    CONTINUE

9      FORMAT(A28,I4,A5)
10     FORMAT(A80)
11     FORMAT(A)
12     FORMAT(A1)
13     FORMAT(A34,I3,A7)
14     FORMAT(A10,I2,A3,I3,A5,F12.10,A6)
16     FORMAT(A50,I3,A4)
18     FORMAT(A29,I4,A3)
19     FORMAT(' ',10(F5.2,2X))
20     FORMAT(' ',10(2X,I3,2X))
25     FORMAT(A11,I3,A11)
28     FORMAT('1')
29     FORMAT(' ',10(F6.1,1X))
30     FORMAT(A18,F7.0,A35,F7.0)
33     FORMAT(A27,F5.2,A30,F5.2)

      STOP
      END

C
C*****
C
C  PARAMETER INPUT SUBROUTINES
C
      SUBROUTINE PRMTRS

```

```

REAL POSTBW,POSTEZ,SNRDB
INTEGER SEQ(64),BITS,BIT,N,N1,ND,TRIALS
CHARACTER*1 C
LOGICAL AUTO,ADF,FSK,SNR,PRE,POSTA,POSTZ,ZC,ZC4,LIM

C
C*****
C
C SEQUENCE
C
      ENTRY PSEQUENCE(SEQ,BITS)

      TYPE *, ' '
      TYPE 11, 'f'      Test sequence or pseudo-random sequence (T/S)? '
      READ (*,12) C

      IF (C.EQ.'T' .OR. C.EQ.'t') THEN
        DO 100 I= 1,BITS/2
          SEQ(2*I) = 1
          SEQ(2*I-1) = 0
100    CONTINUE
      ELSEIF (C.EQ.'S' .OR. C.EQ.'s') THEN
        DO 150 I=0,32,32
          SEQ(I+1) = 1
          SEQ(I+2) = 0
          SEQ(I+3) = 0
          SEQ(I+4) = 1
          SEQ(I+5) = 1
          SEQ(I+6) = 0
          SEQ(I+7) = 1
          SEQ(I+8) = 0
          SEQ(I+9) = 0
          SEQ(I+10) = 1
          SEQ(I+11) = 0
          SEQ(I+12) = 0
          SEQ(I+13) = 0
          SEQ(I+14) = 0
          SEQ(I+15) = 1
          SEQ(I+16) = 0
          SEQ(I+17) = 1
          SEQ(I+18) = 0
          SEQ(I+19) = 1
          SEQ(I+20) = 1
          SEQ(I+21) = 1
          SEQ(I+22) = 0
          SEQ(I+23) = 1
          SEQ(I+24) = 1
          SEQ(I+25) = 0
          SEQ(I+26) = 0
          SEQ(I+27) = 0
          SEQ(I+28) = 1
          SEQ(I+29) = 1

```

```

        SEQ(I+30) = 1
        SEQ(I+31) = 1
        SEQ(I+32) = 1
150      CONTINUE
      ELSE
        DO 200 I=1,BITS
200      SEQ(I) = 0
      ENDIF

      RETURN

C
C*****
C
C NOISE
C
      ENTRY PNOISE(SNRDB,SNR)

      TYPE *, ' '
      TYPE 11, 'f   Do you want to add noise (Y/N)? '
      READ(*,12) C

      IF (C.EQ.'Y' .OR. C.EQ.'y') THEN
        SNR = .TRUE.
        TYPE 11, 'f   Signal to noise ratio (dB)? '
        READ (*,*) SNRDB
      ELSE
        SNR = .FALSE.
      ENDIF

      RETURN

C
C*****
C
C PREFILTER
C
      ENTRY PPREFILTER(PREBW3,PRE)

      TYPE *, ' '
C      TYPE 11, 'f   Do you want to perform pre-filtering? '
C      READ (*,12) C
      C = 'Y'
      IF (C.EQ.'Y' .OR. C.EQ.'y') THEN
        PRE = .TRUE.
        TYPE 11, 'f   Pre-filter bandwidth? '
        READ (*,*) RFBW
        PREBW3 = RFBW/2.
      ELSE
        PRE = .FALSE.
        PREBW3 = 0.0
      ENDIF

      RETURN

```

```

C
C*****
C
C POST FILTER
C
      ENTRY PPOSTFILTER(POSTBW,POSTA,ZC4,POSTBZ)

      TYPE *, ' '
      TYPE 11, 'f Do you want to perform post-filtering? '
      READ (*,12) C
      IF (C.EQ.'Y' .OR. C.EQ.'y') THEN
        POSTA = .TRUE.
        TYPE 11, 'f Post-filter bandwidth? '
        READ (*,*) POSTBW
        POSTBZ = POSTBW
      ELSE
        POSTA = .FALSE.
        IF(ZC4) THEN
          TYPE 11, 'f Post-filter bandwidth for ZC algorithm? '
          READ (*,*) POSTBZ
        ENDIF
      ENDIF
      RETURN

C
C*****
C
C LIMITER
C
      ENTRY PLIM(LIM)

      TYPE *, ' '
      TYPE 11, 'f Do you want to enable the limiter? '
      READ(*,12) C

      IF(C.EQ.'Y' .OR. C.EQ.'y') THEN
        LIM = .TRUE.
      ELSE
        LIM = .FALSE.
      ENDIF
      RETURN

C
C*****
C
C DLD DELAY
C
      ENTRY PDELAY(ND)

      TYPE *, ' '
      TYPE 11, 'f Delay factor of DLD (an integer)? '
      READ(*,*) ND

```

```

      RETURN
C
C*****
C
C FSK FREQUENCY SHIFT
C
      ENTRY PFSKSHIFT(FW)

      TYPE *, ' '
      TYPE 11, 'f' FSK frequency shift? '
      READ (*,*) FW

      RETURN
C
C*****
C
C NUMBER OF TRIALS
C
      ENTRY PTRIALS(TRIALS)

      TYPE *, ' '
      TYPE 11, 'f'Number of trials per run? '
      READ (*,*) TRIALS

      RETURN
C
C*****
C
C TRADITIONAL ZERO CROSSING ENABLE
C
      ENTRY PZCROSS(ZC)

      TYPE *, ' '
      TYPE 11, 'f'Traditional zero crossing algorithm enabled? (Y/N) '
      READ(*,12) CANS
      IF (CANS.EQ.'Y' .OR. CANS.EQ.'y') THEN
         ZC = .TRUE.
      ELSE
         ZC = .FALSE.
      ENDIF

      RETURN
C
C*****
C
C ADAPTIVE NOISE CANCELLING ENABLE
C
      ENTRY PADF(ADF)

      TYPE *, ' '

```

```

TYPE 11, '/Adaptive noise cancelling enabled? (Y/N) '
READ(*,12) CANS
IF (CANS.EQ.'Y' .OR. CANS.EQ.'y') THEN
    ADF = .TRUE.
ELSE
    ADF = .FALSE.
ENDIF

RETURN

C
C*****
C
C AUTOCORRELATION ENABLE
C
    ENTRY PAUTO(AUTO)

    TYPE *, ' '
    TYPE 11, '/Autocorrelation enabled? (Y/N) '
    READ(*,12) CANS
    IF (CANS.EQ.'Y' .OR. CANS.EQ.'y') THEN
        AUTO = .TRUE.
    ELSE
        AUTO = .FALSE.
    ENDIF

    RETURN

C
C*****
C
C FSK SIGNAL INFORMATION ENABLE
C
    ENTRY PFSKINFO(FSK)

    TYPE *, ' '
    TYPE 11, '/Phase and frequency of FSK signal enabled? (Y/N) '
    READ(*,12) CANS
    IF (CANS.EQ.'Y' .OR. CANS.EQ.'y') THEN
        FSK = .TRUE.
    ELSE
        FSK = .FALSE.
    ENDIF

    RETURN

11  FORMAT(A)
12  FORMAT(A1)

END

C
C*****
C
C NOISE GENERATION SUBROUTINE

```

C

```
SUBROUTINE SNOISE(CNOISE,N,A,TS,PBWB,SNRDB,LSNR)
```

```
COMPLEX CNOISE(2048)
```

```
REAL A,TS,PBWB,SNRDB,VAR,MEAN,SNR,RNOISE,INOISE
```

```
INTEGER N
```

```
LOGICAL LSNR
```

```
IF(.NOT.LSNR) THEN
```

```
DO 300 I=1,N
```

```
300 CNOISE(I) = CMPLX(0.,0.)
```

```
ELSE
```

```
SNR = 10.**(SNRDB/10.0)
```

```
VAR = (A**2/2)/(TS*PBWB*SNR)
```

```
MEAN = 0.0
```

```
DO 400 I=1,N
```

```
RNOISE = GAUSS(MEAN,VAR)
```

```
INOISE = GAUSS(MEAN,VAR)
```

```
CNOISE(I) = CMPLX(RNOISE,INOISE)
```

```
400 CONTINUE
```

```
ENDIF
```

```
RETURN
```

```
END
```

C

```
C*****
```

C

```
C PLOT TITLE FORMATION SUBROUTINE
```

C

```
SUBROUTINE PTITLE(TITLE,PRE,POSTA,FW,PBWB,POSTBW,SNRDB,SNR,R,LIM)
```

```
IMPLICIT NONE
```

```
REAL FW,PBWB,POSTBW,R,SNRDB
```

```
CHARACTER TITLE*80,PRETI*13,POSTTI*14
```

```
CHARACTER STI*14,FTI*12,SNRTI*10,LTI*9
```

```
LOGICAL SNR,PRE,POSTA,LIM
```

```
WRITE (PRETI,19) ' , PRE-BW=',IFIX(PBWB*2.)
```

```
19 FORMAT(A,I4)
```

```
WRITE(POSTTI,20) ' , POST-BW=', IFIX(POSTBW)
```

```
20 FORMAT (A,I4)
```

```
WRITE(STI,21) 'BAUD RATE=',IFIX(R)
```

```
21 FORMAT(A10,I4)
```

```
WRITE(FTI,23) ' , SHIFT=',IFIX(FW)
```

```
23 FORMAT(A,I4)
```

```
WRITE (SNRTI,24) ' , SNRDB=',IFIX(SNRDB)
```

```
24 FORMAT(A,I2)
```

```
IF(.NOT.SNR) SNRTI = ' , NO NOISE'
```

```
IF(LIM) THEN
```

```
      LTI = ', LIMITER'
ELSE
      LTI = ' '
ENDIF

IF (PRE) THEN
  IF (POSTA) THEN
    TITLE = STI//FTI//PRETI//POSTTI//SNRTI//LTI
  ELSE
    TITLE = STI//FTI//PRETI//', NO POST'//SNRTI//LTI
  ENDIF
ELSE
  IF (POSTA) THEN
    TITLE = STI//FTI//',NO PRE'//POSTTI//SNRTI//LTI
  ELSE
    TITLE = STI//FTI//',NO PRE, NO POST'//SNRTI//LTI
  ENDIF
ENDIF

RETURN
END
```

```

C*****
C
C      SADF
C
C      VAX-11 FORTRAN SOURCE FILENAME:      SADF1.FOR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING  KANSAS STATE UNIVERSITY
C
C      REVISION      DATE      PROGRAMMER(S)
C      1.0          MAY 6, 1985      MARC D. BRACK
C*****
C
C      CALLING SEQUENCE
C
C          CALL SADF(X,XNN,LENGTH,TS,TITLE)
C
C      PURPOSE
C
C          Implements the adaptive noise canceller in a user
C          controlled environment
C
C      ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
C
C          SPLOT
C          BUTER
C          DFT
C
C      ROUTINE(S) ACCESSED OR CALLED BY EMBEDDED ROUTINE(S)
C
C          NONE
C
C      ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C          X          Output of the delay-line discriminator (DLD)
C
C          XNN        Output of the DLD when the input was noise free
C
C          LENGTH     Number of samples in DLD output
C
C          TS         Sampling interval
C
C          TITLE      Major title for plotting purposes
C
C      ARGUMENT(S) SUPPLIED TO THE CALLING ROUTINE
C
C          NONE
C*****
C
C      SUBROUTINE SADF(X,XNN,LENGTH,TS,TITLE)

```

```

IMPLICIT NONE
INTEGER N,I,NUM1,LENGTH,DELAY,N2,NN,PNUM,P2,TN,TDELAY,TMUI
INTEGER PFIRST,CHOICE,CS1,CS2,J,N3
REAL MU,W(2049,0:256),WMEAN(0:256),WVAR(0:256),WMEANS(0:256)
REAL EROR(2048),LEFT(2048),RIGHT(2048),Y(2048),X(2048)
REAL FEROR(2048),FY(2048),BW,MAXF,FX(2048)
REAL WDFT(4096),TMU,TS,FREQP,XNN(2048),XSCALE
COMPLEX CW(2048),CW2(2048),BUTER
CHARACTER*80 NAME1,NAME2,BNK,NAME,LT,RT,HT,TITLE,STITLE,HU
CHARACTER DELAYC*4,NC*4,MUC*11,WAIT*1,CANS*1,CANS2*1
LOGICAL WBIAS,FILT

```

```

C
C INPUT PARAMETERS FOR ROUTINE
C

```

```

      CALL PADFDELAY(DELAY)
      CALL PWEIGHTS(N)
      CALL PCONSTANT(MU)
      CALL PBIAS(WBIAS)
      CALL PFILTER(FILT,BW)

```

```

C
C*****
C

```

```

C ADAPTIVE NOISE CANCELLER SECTION, LMS ALGORITHM
C

```

```

1000    CONTINUE

```

```

      DO 100 I=0,N
        WMEAN(I) = 0.
        WMEANS(I) = 0.
        WVAR(I) = 0.

```

```

100    CONTINUE

```

```

      DO 120 J=1,LENGTH+1

```

```

        DO 110 I=0,N
          W(J,I) = 0.0

```

```

110    CONTINUE
120

```

```

      DO 160 J=1,LENGTH
        IF (WBIAS) THEN
          Y(J) = W(J,0)
        ELSE
          Y(J) = 0.0
        ENDIF

```

```

      DO 130 I=1,N
        IF(J-I+1.GT.0) Y(J) = Y(J) + X(J-I+1)*W(J,I)

```

```

130
      IF(J+DELAY.LE.LENGTH) THEN
        EROR(J) = X(J+DELAY) - Y(J)
      ELSE

```

```

        EROR(J) = -Y(J)
    ENDIF

    W(J+1,0) = W(J,0) + 2.*MU*EROR(J)

    DO 140 I=1,N
140      IF(J-I+1.GT.0) W(J+1,I) = W(J,I) + 2.*MU*EROR(J)*X(J-I+1)

        IF(J.GT.N) THEN
            DO 150 I=0,N
                WMEAN(I) = WMEAN(I) + W(J+1,I)/LENGTH
                WMEANS(I) = WMEANS(I) + W(J+1,I)**2/LENGTH
150            CONTINUE
            ENDIF
160        CONTINUE

        DO 170 I=0,N
170          WVAR(I) = WMEANS(I) - WMEAN(I)**2

C
C  FILTER THE ERROR AND THE FILTER OUTPUT
C
        IF (FILT) THEN
            DO 180 I=0,N
180              CW(I+1) = CMPLX(WMEAN(I),0.)

                DO 190 I=N+1,4*N-1
190                  CW(I+1) = CMPLX(0.,0.)
                C
C  DETERMINE 3 dB BANDWIDTH OF SIGNAL
C
                CALL DFT(CW,4*N,0)

                MAXF = 0
                DO 200 I=1,4*N/2
200                  IF(CABS(CW(I)).GT.MAXF) MAXF = CABS(CW(I))
                MAXF = MAXF**2

                DO 210 I=1,4*N/2
                    IF(CABS(CW(4*N/2+1-I))**2.GT.MAXF/2) THEN
                        BW = (4*N/2+2-I)/(4*N/2*TS)
                        GOTO 220
                    ENDIF
210                CONTINUE
220                CONTINUE

                TYPE *, 'BANDWIDTH = ',BW
            ENDIF

C
C*****
C
C  PLOTTING AND PARAMETER CHANGE

```

```

C
3000 WRITE(*,3)
3   FORMAT(' Do you want to plot:',/
      ' 1. DLD output',/
      ' 2. Output error',/
      ' 3. Filter estimate',/
      ' 4. Weights of filter for a specific vector',/
      ' 5. Frequency response of filter',/
      ' 6. Mean and variance of weights',/
      ' 7. History of a specific weight',/
      ' Or do you want to:',/
      ' 8. Change the input delay',/
      ' 9. Change the gradient constant',/
      '10. Change the number of weights',/
      '11. Change the weighting bias',/
      '12. Write the results to disk',/
      '13. Change the automatic filter enable',/
      '14. Perform a run',/
      '15. Return to the calling routine')
      TYPE 1, 'Enter selection: '
      READ (*,*) CHOICE

      IF (CHOICE.LE.7) THEN
1         FORMAT(A)
2         FORMAT(A1)
          WRITE(*,*) ' '
          TYPE 1, 'First point to plot = '
          READ (*,*) PFIRST
C         TYPE 1, 'Interval between points = '
C         READ (*,*) NN
          ENDIF

          NN = 1

          GOTO (3100,3150,3200,3300,3400,3500,3600,4000,
X           4100,4200,4300,4400,4500,1000,9000), CHOICE
C*****
C
C PLOTTING
C
C PLOT DLD OUTPUT
C
3100  TYPE 1, 'Number of points to plot? '
      READ (*,*) PNUM
      IF (FILT) THEN
          TYPE 1, 'Plot DLD output or filtered DLD output? (D/F) '
          READ (*,2) CANS
      ELSE
          CANS = 'D'
      ENDIF
C

```

```

C   PERFORM INTERMEDIATE FILTERING
C
      IF (CANS.EQ.'F' .OR. CANS.EQ.'f') THEN
        DO 290 J=1,LENGTH
290          CW(J) = CMPLX(X(J),0.)

          CALL DFT(CW,LENGTH,0)
          DO 300 J=1,LENGTH/2
            CW(J) = CW(J)*BUTER(4,BW,FLOAT(J-1)/(TS*LENGTH))
            CW(LENGTH+1-J) = CW(LENGTH+1-J)*
X              BUTER(4,BW,FLOAT(-J)/(TS*LENGTH))
300          CONTINUE
            CW(LENGTH/2+1) = CMPLX(0.,0.)
            CALL DFT(CW,LENGTH,1)

            DO 310 J=1,LENGTH
310              FX(J) = REAL(CW(J))
            ENDIF

            DO 3110 I=PFIRST,LENGTH
              IF(I+DELAY.LT.LENGTH) THEN
                RIGHT(I-PFIRST+1) = XNN(I+DELAY)
              ELSE
                RIGHT(I-PFIRST+1) = 0.0
              ENDIF
              IF (CANS.EQ.'F' .OR. CANS.EQ.'f') THEN
                LEFT(I-PFIRST+1) = FX(I)
              ELSE
                LEFT(I-PFIRST+1) = X(I)
              ENDIF
3110            CONTINUE

            RT = 'NOISELESS SIGNAL'
            IF (CANS.EQ.'F'.OR.CANS.EQ.'f') THEN
              LT = 'FILTERED DLD OUTPUT'
            ELSE
              LT = 'DLD OUTPUT'
            ENDIF
            HT = 'SAMPLE NUMBER'
            HU = ' '
            XSCALE = 1
C          PNUM = LENGTH - PFIRST + 1
          GOTO 3900
C
C   PLOT ERROR OF ANC
C
3150  TYPE 1, '/Number of points to plot? '
      READ (*,*) PNUM
      IF (FILT) THEN
        TYPE 1, '/Plot error or filtered error? (E/F) '
        READ (*,2) CANS
      ELSE

```

```

      CANS = 'E'
    ENDIF

C   PERFORM INTERMEDIATE FILTERING
C
    IF (CANS.EQ.'F' .OR. CANS.EQ.'f') THEN
      DO 260 J=1,LENGTH
260      CW(J) = CMPLX(EROR(J),0.)

      CALL DFT(CW,LENGTH,0)
      DO 270 J=1,LENGTH/2
        CW(J) = CW(J)*BUTER(4,BW,FLOAT(J-1)/(TS*LENGTH))
        CW(LENGTH+1-J) = CW(LENGTH+1-J)*
X      BUTER(4,BW,FLOAT(-J)/(TS*LENGTH))
270      CONTINUE
      CW(LENGTH/2+1) = CMPLX(0.,0.)
      CALL DFT(CW,LENGTH,1)

      DO 280 J=1,LENGTH
280      FEROR(J) = REAL(CW(J))
    ENDIF

    DO 3160 I=PFIRST,LENGTH
      IF(I+DELAY.LT.LENGTH) THEN
        RIGHT(I-PFIRST+1) = XNN(I+DELAY)
      ELSE
        RIGHT(I-PFIRST+1) = 0.0
      ENDIF
      IF (CANS.EQ.'F' .OR. CANS.EQ.'f') THEN
        LEFT(I-PFIRST+1) = FEROR(I)
      ELSE
        LEFT(I-PFIRST+1) = EROR(I)
      ENDIF
3160    CONTINUE

    RT = 'NOISELESS SIGNAL'
    IF (CANS.EQ.'F'.OR.CANS.EQ.'f') THEN
      LT = 'FILTERED ERROR'
    ELSE
      LT = 'ERROR'
    ENDIF
    HT = 'SAMPLE NUMBER'
    HU = ' '
    XSCALE = 1
C   PNUM = LENGTH - PFIRST + 1
    GOTO 3900

C
C   PLOT ADF OUTPUT
C
3200  TYPE 1, '/Number of points to plot? '
      READ (*,*) PNUM
      IF (FILT) THEN

```

```

        TYPE 1, '/Plot filter output or filtered filtered output? (0/F) '
        READ (*,2) CANS
    ELSE
        CANS = '0'
    ENDIF

C
C PERFORM INTERMEDIATE FILTERING
C
    IF (CANS.EQ.'F' .OR. CANS.EQ.'f') THEN
        DO 230 J=1,LENGTH
230      CW(J) = CMPLX(Y(J),0.)

        CALL DFT(CW,LENGTH,0)
        DO 240 J=1,LENGTH/2
            CW(J) = CW(J)*BUTER(4,BW,FLOAT(J-1)/(TS*LENGTH))
            CW(LENGTH+1-J) = CW(LENGTH+1-J)*
X          BUTER(4,BW,FLOAT(-J)/(TS*LENGTH))
240      CONTINUE
        CW(LENGTH/2+1) = CMPLX(0.,0.)
        CALL DFT(CW,LENGTH,1)

        DO 250 J=1,LENGTH
250      FY(J) = REAL(CW(J))
        ENDIF

        DO 3210 I=PFIRST,LENGTH
            IF(I+DELAY.LT.LENGTH) THEN
                RIGHT(I-PFIRST+1) = XNN(I+DELAY)
            ELSE
                RIGHT(I-PFIRST+1) = 0.0
            ENDIF
            IF (CANS.EQ.'F' .OR. CANS.EQ.'f') THEN
                LEFT(I-PFIRST+1) = FY(I)
            ELSE
                LEFT(I-PFIRST+1) = Y(I)
            ENDIF
3210      CONTINUE

        RT = 'NOISELESS SIGNAL'
        IF (CANS.EQ.'F' .OR. CANS.EQ.'f') THEN
            LT = 'FILTERED FILTER OUTPUT'
        ELSE
            LT = 'FILTER OUTPUT'
        ENDIF
        HT = 'SAMPLE NUMBER'
        HU = ' '
        XSCALE = 1
C      PNUM = LENGTH - PFIRST + 1
        GOTO 3900

C
C PLOT A SPECIFIC WEIGHT VECTOR
C

```

```

3300 IF(PFIRST.GT.N) THEN
      WRITE(*,*) 'ERROR: THERE ARE NOT THAT MANY WEIGHTS'
      GOTO 3000
ENDIF
WRITE(*,*) ' '
TYPE 1, '/Weights for what vector number? (a,b) '
READ (*,*) CS1,CS2
IF (CS2.GE.1) THEN
      DO 3310 I=0,N
            LEFT(I-PFIRST+2) = W(CS1,I-PFIRST+1)
            RIGHT(I-PFIRST+2) = W(CS2,I-PFIRST+1)
3310 CONTINUE
      WRITE(RT,8) 'VECTOR NUMBER ',CS2
8      FORMAT(A17,I4)
      ELSE
            DO 3320 I=0,N
3320 LEFT(I-PFIRST+2) = W(CS1,I-PFIRST+1)
            RT = 'NONE'
      ENDIF

      WRITE(LT,8) 'VECTOR NUMBER ',CS1
      HT = 'WEIGHT NUMBER'
      HU = ' '
      NN = 1
      XSCALE = 1
      PNUM = N+1
      GOTO 3900

C
C PLOT FREQUENCY RESPONSE OF ADF
C
3400 N2 = 1
3410 N2 = N2*2
      IF(N2.LT.N+1) GOTO 3410

      WRITE(*,*) ' '
      WRITE(*,6) N+1,' points, need to zero pad to at least ',N2,' points'
6      FORMAT(I4,A,I4,A)
      TYPE 1, '/Number of points to zero pad to = '
      READ (*,*) N3

      N2 = 1
      N2 = N2*2
3415 IF(N2.LT.N3) GOTO 3415
      WRITE(*,*) 'Zero padded to ',N2,' points'

      WRITE (*,*) ' '
      WRITE (*,9)
9      FORMAT (' Frequency response desired for: ',/,
X      ' 0. Mean and variance of weights',/,
X      ' #. Specific vector of weights')
      TYPE 1, '/Enter 0 or vector number: '

```

```

READ (*,*)CS1

IF (CS1.NE.0) THEN
  DO 3420 I=0,N
    3420    CW(I+1) = CMPLX(W(CS1,I),0.0)
    IF(N2.GT.N+1) THEN
      DO 3430 I=N+1,N2-1
        3430    CW(I+1) = CMPLX(0.,0.)
      ENDIF
    ELSE
      DO 3440 I=0,N
        CW(I+1) = CMPLX(WMEAN(I),0.0)
        CW2(I+1) = CMPLX(WVAR(I),0.0)
      3440    CONTINUE
      IF(N2.GT.N+1) THEN
        DO 3450 I=N+1,N2-1
          CW(I+1) = CMPLX(0.,0.)
          CW2(I+1) = CMPLX(0.,0.)
        3450    CONTINUE
        ENDIF
      ENDIF
    ENDIF

    CALL DFT(CW,N2,0)
    CALL DFT(CW2,N2,0)

    DO 3460 I=1,N2/2
      LEFT(I) = CABS(CW(I))
      RIGHT(I) = CABS(CW2(I))
    3460    CONTINUE

    IF(PFIRST.GT.N2/2) THEN
      WRITE(*,*) 'THERE ARE ONLY ',N2/2,' RESPONSES'
      GOTO 3000
    ENDIF

    IF (CS1.EQ.0) THEN
      LT = 'RESPONSE OF MEAN'
      RT = 'RESPONSE OF VARIANCE'
    ELSE
      WRITE(LT,10) 'RESPONSE OF VECTOR ',CS1
    10    FORMAT(A19,I4)
      RT = 'NONE'
    ENDIF

    HT = 'FREQUENCY RESPONSE'
    HU = 'HERTZ'
    XSCALE = 1./(N2*TS)
    WRITE(*,*) ' '
    TYPE 1,'Plot up to what frequency? (4800 maximum) '
    READ (*,*) FREQP
    PNUM = FREQP/XSCALE
    GOTO 3900

```

C PLOT MEAN AND VARIANCE OF WEIGHTS

C

```

3500   DO 3510 I=0,N
        LEFT(I+1) = WMEAN(I)
        RIGHT(I+1) = WVAR(I)
3510   CONTINUE
        LT = 'MEAN OF WEIGHTS'
        RT = 'VARIANCE OF WEIGHTS'
        HT = 'WEIGHT NUMBER'
        HU = ' '
        XSCALE = 1.
        PNUM = N+1
        GOTO 3900

```

C

C PLOT ONE WEIGHT THROUGHOUT THE UPDATE PROCEDURE

C

```

3600   TYPE 1, 'fHistory of what number weights? (enter a,b) '
        READ (*,*) CS1,CS2
        IF(CS2.GE.0) THEN
            DO 3610 J=1,LENGTH
                LEFT(J) = W(J,CS1)
                RIGHT(J) = W(J,CS2)
3610   CONTINUE
            WRITE(RT,7) 'CROSS-SECTION NUMBER ',CS2
7       FORMAT(A21,I4)
        ELSE
            DO 3620 J=1,LENGTH
3620   LEFT(J) = W(J,CS1)
            RT = 'NONE'
        ENDIF

        WRITE(LT,7) 'CROSS-SECTION NUMBER ',CS1
        HT = 'WEIGHT CROSS-SECTION'
        HU = ' '
        PNUM = J
        XSCALE = TS
        GOTO 3900

```

3900 CONTINUE

C

C FORM SUB-TITLE AND CALL PLOTTING ROUTINE

C

```

        BNK = ' '
        IF (WBIAS) THEN
            WRITE(STITLE,5) 'ADF-DELAY=',DELAY,', N=',N,', MU=',MU,', BIAS'
5       FORMAT(A10,I2,A4,I4,A5,F12.10,A6)
        ELSE
            WRITE(STITLE,5) 'ADF-DELAY=',DELAY,', N=',N,', MU=',MU
        ENDIF
        CALL SPLOT(LEFT,RIGHT,PNUM,NN,LT,BNK,RT,BNK,HT,
X       HU,TITLE,STITLE,XSCALE)
        READ(*,2) WAIT

```

```

      GOTO 3000
C
C*****
C
C  PARAMETER CHANGE
C
4000  CALL PADFDELAY(DELAY)
      GOTO 3000

4100  CALL PCONSTANT(MU)
      GOTO 3000

4200  CALL PWEIGHTS(N)
      GOTO 3000

4300  CALL PBIAS(WBIAS)
      GOTO 3000

4400  WRITE(*,*) 'NOT AVAILABLE AT THIS TIME'
      GOTO 3000

4500  CALL PFILTER(FILT,BW)
      GOTO 3000

9000  CONTINUE

      RETURN
      END
C
C
C
C*****
C
C  THE FOLLOWING SUBROUTINES ARE QUERIES FOR THE ADAPTIVE NOISE
C  CANCELLER PARAMETERS.  THEY ARE USED IN BOTH THE USER
C  CONTROLLED ANC SUBROUTINE AND THE MAIN SIMULATION ROUTINE.
C
      SUBROUTINE PARAMETERS
      REAL MU,BW
      INTEGER DELAY,N
      CHARACTER CANS*1
      LOGICAL WBIAS,FILT,OFILT,SZC,SWV,SMM,SCORR,SANCZC

1     FORMAT(A)
C
C  NUMBER OF WEIGHTS OF WEIGHT VECTOR
C
      ENTRY PWEIGHTS(N)
      WRITE(*,*) ' '
      TYPE 1, 'f'   Number of weights = '
      READ (*,*) N

```

```

      RETURN
C
C  DELAY OF INPUT TO ADAPTIVE FILTER
C
      ENTRY PADFDELAY(DELAY)
      WRITE(*,*) ' '
      TYPE 1, 'f'   Reference input delay = '
      READ (*,*) DELAY
      RETURN
C
C  RATE OF CONVERGENCE FACTOR
C
      ENTRY PCONSTANT(MU)
      WRITE(*,*) ' '
      TYPE 1, 'f'   Gradient proportionality constant = '
      READ (*,*) MU
      RETURN
C
C  WEIGHT BIAS ENABLING
C
      ENTRY PBIAS(WBIAS)
      WRITE(*,*) ' '
      TYPE 1, 'f'   Do you want to include bias for weights? (Y/N) '
      READ (*,2) CANS
2      FORMAT(A1)
      IF (CANS.EQ.'Y'.OR.CANS.EQ.'y') THEN
         WBIAS = .TRUE.
      ELSE
         WBIAS = .FALSE.
      ENDIF
      RETURN
C
C  BANDWIDTH ESTIMATOR FOR USER CONTROLLED ANC
C
      ENTRY PFILTER(FILT,BW)
      WRITE(*,*) ' '
      TYPE 1, 'f'   Do you want to make a bandwidth estimate (Y/N) '
      READ (*,2) CANS
      IF (CANS.EQ.'Y'.OR.CANS.EQ.'y') THEN
         FILT = .TRUE.
      ELSE
         FILT = .FALSE.
      ENDIF
      RETURN
C
C  INTERMEDIATE FILTERING ENABLE FOR USER CONTROLLED ANC
C
      ENTRY PFILT1(OFILT)
      WRITE(*,*) ' '
      TYPE 1, 'f'   Do you want to enable the intermediate filter? (Y/N) '
      READ (*,2) CANS
      IF (CANS.EQ.'Y'.OR.CANS.EQ.'y') THEN

```

```

      OFILT = .TRUE.
    ELSE
      OFILT = .FALSE.
    ENDIF
    RETURN
C
C  ENABLE ESTIMATES BASED ON THE WEIGHT VECTOR
C
      ENTRY PMM(SMM)
      WRITE(*,*) ' '
      TYPE 1, 'f'      Do you want to enable the Max/Min estimators? (Y/N) '
      READ (*,2) CANS
      IF (CANS.EQ.'Y'.OR.CANS.EQ.'y') THEN
        SMM = .TRUE.
      ELSE
        SMM = .FALSE.
      ENDIF
      RETURN
C
C  ENABLE ESTIMATES BASED ON SIMILARITIES BETWEEN WEIGHT VECTOR AND
C  AUTOCORRELATION ESTIAMTE
C
      ENTRY PCORR(SCORR)
      WRITE(*,*) ' '
      TYPE 3, 'f'      Do you want to enable the',
X      'f pseudo-corr. estimators? (Y/N) '
3      FORMAT(A,A)
      READ (*,2) CANS
      IF (CANS.EQ.'Y'.OR.CANS.EQ.'y') THEN
        SCORR = .TRUE.
      ELSE
        SCORR = .FALSE.
      ENDIF
      RETURN
C
C  ENABLE ESTIMATES BASED ON ZERO CROSSING ANALYSIS
C
      ENTRY PANCZC(SANCZC)
      WRITE(*,*) ' '
      TYPE 1, 'f'      Do you want to enable the ANC-ZC estimators? (Y/N) '
      READ (*,2) CANS
      IF (CANS.EQ.'Y'.OR.CANS.EQ.'y') THEN
        SANCZC = .TRUE.
      ELSE
        SANCZC = .FALSE.
      ENDIF
      RETURN
END

```

 C
 C
 C SSADF
 C
 C
 C VAX-11 FORTRAN SOURCE FILENAME: SADF3
 C
 C
 C DEPARTMENT OF ELECTRICAL ENGINEERING KANSAS STATE UNIVERSITY
 C
 C
 C REVISION DATE PROGRAMMER(S)
 C
 C 0.0 MAY 6, 1985 MARC D. BRACK
 C
 C *****

C
 C
 C CALLING SEQUENCE
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C

CALL SSADF(X,LENGTH,MU,N,DELAY,WBIAS,WMEAN,WVAR,
 FILT,SZEST,TS,MEST,BW,SMM,
 ERORI,YI,EROR,Y,SCORR,SANCZC)

C
 C
 C PURPOSE
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C

This subroutine implements the Adaptive Noise
 Canceller Baud Rate Estimator. The routine is
 to be used inconjunction with SBAUD.

C
 C
 C ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C

DFT
 BUTER
 ZCROSS
 BITLENGTH

C
 C
 C ROUTINE(S) ACCESSED OR CALLED BY EMBEDDED ROUTINE(S)
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C

NONE

C
 C
 C ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C

X Array of input sequence (real)
 LENGTH Number of elements of X (integer)
 MU Gradient constant for LMS algorithm (real)
 N Number of weights in weight vector (integer)
 DELAY Number of samples for delay (integer)
 WBIAS Flag for weight bias, TRUE if bias
 is to be included (logical)

```

C          FILT      Flag for intermediate filtering, TRUE if
C                   filtering to be done (logical)
C
C          TS        Sampling interval of inpnt sequence (real)
C
C          SMM        Flag for bitlength analysis, TRUE if
C                   analysis to be done (logical)
C
C          SCORR      Flag for pseudo-correlation analysis, TRUE
C                   if analysis to be done (logical)
C
C          SANCZC     Flag for zero crossing analysis, TRUE if
C                   analysis to be done (logical)
C
C ARGUMENT(S) SUPPLIED TO THE CALLING ROUTINE
C
C          SZEST      Array for estimates made with secondary
C                   adaptive noise canceller (real)
C
C          MMEST      Array for estimates made with intermediate
C                   adaptive noise canceller (real)
C
C          BW         Bandwidth estimate of demodulator output (real)
C
C          X          Array of intermediately filtered demodulator
C                   output (real)
C
C          ERORI      Array of error sequence from intermediate
C                   adaptive noise canceller (real)
C
C          YI         Array of adaptive filter output sequence from
C                   intermediate adaptive noise canceller (real)
C
C          EROR       Array of error sequence from secondary
C                   adaptive noise canceller (real)
C
C          Y          Array of adaptive filter output sequence from
C                   secondary adaptive noise canceller (real)
C
C          WMEAN      Array of mean of filter weights (real)
C
C          WVAR       Array of variance of filter weights (real)
C
C *****
C

```

```

SUBROUTINE SSADF(X,LENGTH,MU,N,DELAY,WBIAS,WMEAN,WVAR,
X          FILT,SZEST,TS,MMEST,BW,SMM,
X          ERORI,YI,EROR,Y,SCORR,SANCZC)

```

```

IMPLICIT NONE

```

```

INTEGER N,I,J,LENGTH,DELAY,NUM,NUM2,MINJM,MINJV
INTEGER HFLAG(10),HISTA(2,10,257),MAX,ACCUM

```

```

REAL MU,W(0:256),WMEAN(0:N),WVAR(0:N),WMEANS(0:256),MAXF
REAL X(LENGTH),SZEST(10),BW,MEST(10),MINF,R(512)
REAL Y(2048),EROR(2048),TS,SBITL(10),TEMPMU,L(512)
REAL YI(2048),ERORI(2048)
COMPLEX CW(2048),BUTER
LOGICAL SMM,FILT,WBIAS,TWO,INTERM,SCORR,SANCZC
CHARACTER*80 B

```

```

INTERM = .FALSE.
TEMPMU = MU
IF(FILT) MU = 0.000000001

```

```

C
C*****
C
C ADAPTIVE NOISE CANCELLING SECTION
C
1000 CONTINUE

      DO 100 I=0,N
        WMEAN(I) = 0.
        WMEANS(I) = 0.
        W(I) = 0.0
100 CONTINUE

C
C LMS IMPLEMENTATION OF ANC
C
      DO 160 J=1,LENGTH
        IF (WBIAS) THEN
          Y(J) = W(0)
        ELSE
          Y(J) = 0.0
        ENDIF

      DO 130 I=1,N
130 IF(J-I+1.GT.0) Y(J) = Y(J) + X(J-I+1)*W(I)

        IF(J+DELAY.LE.LENGTH) THEN
          EROR(J) = X(J+DELAY) - Y(J)
        ELSE
          EROR(J) = -Y(J)
        ENDIF

        W(0) = W(0) + 2.*MU*EROR(J)

      DO 140 I=1,N
140 IF(J-I+1.GT.0) W(I) = W(I) + 2.*MU*EROR(J)*X(J-I+1)

C
C END OF LMS STRUCTURE
C
C CALCULATE MEAN AND MEAN SQUARE OF WEIGHTS
C

```

```

      IF(J.GT.N) THEN
        DO 150 I=0,N
          WMEAN(I) = WMEAN(I) + W(I)/(LENGTH-N)
          WMEANS(I) = WMEANS(I) + W(I)**2/(LENGTH-N)
150      CONTINUE
        ENDIF

160      CONTINUE
      C
      C END OF LMS ALGORITHM IMPLEMENTATION LOOP
      C
        DO 170 I=0,N
170      WVAR(I) = WMEANS(I) - WMEAN(I)**2
      C
      C END OF ANC IMPLEMENTATION
      C
      C*****
      C
      C OBTAIN PSEUDO-CORRELATION ESTIMATES
      C
        IF(SCORR) THEN
          DO 5620 J=0,N
            CW(J+1) = CMPLX(WMEAN(N-J),0.) * FLOAT(J)/N
5620      CONTINUE
          DO 5625 J=N+1,2*N
            CW(J+1) = CMPLX(WMEAN(J-N),0.) * (1.0 - FLOAT(J-N)/N)
5625      CONTINUE
          DO 5630 J=2*N+1,4*N-1
5630      CW(J+1) = CMPLX(0.,0.)
          CALL DFT(CW,4*N,0)
          MINF = 10000000
          DO 5640 J=1,2*N
            L(J) = 20*ALOG10(CABS(CW(J)))
            IF(CABS(CW(J)).LT.MINF) THEN
              MINFM = J
              MINF = CABS(CW(J))
            ENDIF
5640      CONTINUE

          DO 5660 J=0,N
            CW(J+1) = CMPLX(WVAR(N-J),0.) * FLOAT(J)/N
5660      CONTINUE
          DO 5665 J=N+1,2*N
            CW(J+1) = CMPLX(WVAR(J-N),0.) * (1. - FLOAT(J-N)/N)
5665      CONTINUE
          DO 5670 J=2*N+1,4*N-1
5670      CW(J+1) = CMPLX(0.,0.)
          CALL DFT(CW,4*N,0)
          MINF = 10000000
          DO 5680 J=1,2*N
            R(J) = 20*ALOG10(CABS(CW(J)))
            IF(CABS(CW(J)).LT.MINF) THEN

```

```

        MINJV = J
        MINF = CABS(CW(J))
    ENDIF
5680    CONTINUE
    ENDIF

C
C*****
C
C OBTAIN ESTIMATES BASED ON MINIMUM AND MAXIMUM OF MEAN AND VARIANCE
C AND FROM ZERO CROSSING ANALYSIS
C
    TWO = .TRUE.
    IF(SMM .AND. .NOT.INTERM) THEN
        CALL BITLENGTH(WMEAN,WVAR,N,DELAY,MEST,TWO)
        MEST(6) = FLOAT(2*(4*N))/MINJM
        MEST(7) = FLOAT(2*(4*N))/MINJV
        CALL ZCROSS(EROR,LENGTH,MEST(10),1)
    ENDIF

    IF(FILT .AND. INTERM) THEN
        CALL BITLENGTH(WMEAN,WVAR,N,DELAY,SZEST,TWO)
        SZEST(6) = FLOAT(2*(4*N))/MINJM
        SZEST(7) = FLOAT(2*(4*N))/MINJV
        CALL ZCROSS(X,LENGTH,MEST(8),2)
        CALL ZCROSS(Y,LENGTH,SZEST(8),2)
        CALL ZCROSS(EROR,LENGTH,SZEST(10),1)
    ENDIF

C
C*****
C
C IMPLEMENT INTERMEDIATE FILTERING. THE FILTER BANDWIDTH IS BASED
C ON THE 3 dB BANDWIDTH FOUND BY TAKING THE DFT OF THE AVERAGE
C FILTER WEIGHTS.
C
    IF (FILT .AND. .NOT.INTERM) THEN
        INTERM = .TRUE.
        DO 210 I=0,N
210          CW(I+1) = CMPLX(WMEAN(I),0.)

        DO 220 I=N+1,4*N-1
220          CW(I+1) = CMPLX(0.,0.)
        C
        C DETERMINE 3 dB BANDWIDTH OF SIGNAL
        C
        CALL DFT(CW,4*N,0)

        MAXF = 0
        DO 230 I=1,4*N/2
230          IF(CABS(CW(I)).GT.MAXF) MAXF = CABS(CW(I))
        MAXF = MAXF**2

        DO 240 I=1,4*N/2

```

```

        IF(CABS(CW(4*N/2+1-I))**2.GT.MAXF/2) THEN
            BW = (4*N/2+2-I)/(4*N/2*TS)
            GOTO 250
        ENDIF
240      CONTINUE
250      CONTINUE
C
C  PERFORM FILTERING
C
        DO 320 J=1,LENGTH
320      CW(J) = CMPLX(X(J),0.0)

        CALL DFT(CW,LENGTH,0)
        DO 330 J=1,LENGTH/2
            CW(J) = CW(J)*BUTER(4,BW,FLOAT(J-1)/(TS*LENGTH))
            CW(LENGTH+1-J) = CW(LENGTH+1-J)*BUTER(4,BW,FLOAT(-J)/(TS*LENGTH))
330      CONTINUE
            CW(LENGTH/2+1) = CMPLX(0.,0.)
            CALL DFT(CW,LENGTH,1)

            DO 340 J=1,LENGTH
                ERORI(J) = ERROR(J)
                YI(J) = Y(J)
340      X(J) = REAL(CW(J))
C
C  ENTER ANC FOR SPECIFIED MU AFTER INTERMEDIATE FILTERING IS COMPLETE
C
            MU = TEMPMU
            GOTO 1000

        ENDIF

RETURN
END

```

```

*****
C
C      BITLENGTH
C
C      VAX-11 FORTRAN SOURCE FILENAME:          BIT.FOR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING      KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER(S)
C      -----          -
C      0.0              MAY 6, 1985      MARC D. BRACK
C
*****
C
C      CALLING SEQUENCE
C
C      CALL BITLENGTH(WMEAN,WVAR,N,DELAY,BITL,TWO)
C
C      PURPOSE
C      This routine makes estimates of the bad rate by
C      estimating the bit interval in number of sampling
C      intervals. The estimate is taken from the weight
C      vector information.
C
C      ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
C
C      NONE
C
C      ROUTINE(S) ACCESSED OR CALLED BY EMBEDDED ROUTINE(S)
C
C      NONE
C
C      ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C      WMEAN      Array of length N of mean of the
C                  weight vector (real)
C
C      WVAR       Array of length N of variance of the
C                  weight vector (real)
C
C      N          Number of weight of the weight vector (integer)
C
C      DELAY      Delay of the primary input of the ANC (integer)
C
C      TWO        TRUE if the estimates involving the
C                  variance are to be made also (logical)
C
C      ARGUMENT(S) SUPPLIED TO THE CALLING ROUTINE
C
C      BITL       Array containing estimates (real)
C
*****

```

C

```
SUBROUTINE BITLENGTH(WMEAN,WVAR,N,DELAY,BITL,TWO)
```

```
IMPLICIT NONE
```

```
INTEGER MEXT(256),FRMIN,ZC
```

```
INTEGER RI,LI,DELAY,N,I,Q1,Q2,Q3,Q4,Q5,Q6,TEMP,FRMAX,L,R,J
```

```
REAL WMEAN(0:N),WVAR(0:N),BITL(10)
```

```
LOGICAL MINMAX,SIDED1,SIDED2,TWO,NSET,NSET1,NSET2,KEY
```

```
DO 100 I=1,10
```

```
    BITL(I) = 1 - DELAY
```

```
100
```

```
C
```

```
C FIND ALL RELATIVE MINIMA OF THE WEIGHT MEAN FILE AND ALL RELATIVE  
C MAXIMA OF THE WEIGHT VARIANCE FILE
```

```
C
```

```
    NSET = .TRUE.
```

```
    NSET1 = .TRUE.
```

```
    NSET2 = .TRUE.
```

```
    DO 200 I=1,N-1
```

```
        IF(WMEAN(I).LT.WMEAN(I-1) .AND. WMEAN(I).LT.WMEAN(I+1)) THEN
```

```
            MEXT(I) = 1
```

```
            IF(NSET) THEN
```

```
                NSET = .FALSE.
```

```
                FRMIN = I
```

```
            ENDIF
```

```
        ELSEIF(WMEAN(I).GT.WMEAN(I-1) .AND. WMEAN(I).GT.WMEAN(I+1)) THEN
```

```
            MEXT(I) = -1
```

```
            IF(NSET1) THEN
```

```
                NSET1 = .FALSE.
```

```
                FRMAX = I
```

```
            ENDIF
```

```
        ELSE
```

```
            MEXT(I) = 0
```

```
        ENDIF
```

```
        IF(.NOT.NSET1 .AND. NSET2) THEN
```

```
            IF(WMEAN(I)*WMEAN(I+1).LE.0.0) THEN
```

```
                ZC = I
```

```
                NSET2 = .FALSE.
```

```
            ENDIF
```

```
        ENDIF
```

```
200
```

```
    CONTINUE
```

```
C
```

```
C ONE ESTIMATE IS THE FIRST TIME A RELATIVE MAXIMUM AND A RELATIVE  
C MINIMUM OCCUR AT THE SAME POINT. IF THIS DOES NOT HAPPEN DEFAULT  
C TO FIRST RELATIVE MINIMUM PAST THE FIRST RELATIVE MAXIMUM
```

```
C
```

```
    IF (TWO) THEN
```

```
        DO 300 I=FRMAX,N
```

```
            IF(MEXT(I).EQ.1) THEN
```

```

        CALL FMAX(WVAR,I,0,N,KEY)
        IF(KEY) THEN
            BITL(1) = I
            GOTO 310
        ENDIF
    ENDIF
300    CONTINUE
        BITL(1) = FRMIN
310    CONTINUE
C
C  A SECOND ESTIMATE IS IF A VARIANCE MASIMUM IS WITHIN TWO SAMPLES OF
C  THE MEAN MINIMUM
C
        DO 320 I=FRMAX,N
            IF(MEXT(I).EQ.1) THEN
                CALL FMAX(WVAR,I,2,N,KEY)
                IF(KEY) THEN
                    BITL(2) = I
                    GOTO 330
                ENDIF
            ENDIF
320    CONTINUE
        BITL(2) = FRMIN
330    CONTINUE
    ENDIF
C
C  THE THIRD ESTIMATE IS THE ABSOLUTE MINIMUM OF THE MEAN
C
        BITL(3) = FRMAX

        DO 400 I=FRMAX+1,N
            IF(WMEAN(I).LT.WMEAN(BITL(3))) BITL(3) = I
400    CONTINUE
C
C  THE FOURTH ESTIMATE IS THE FIRST RELATIVE MINIMUM LESS THAN ZERO
C  IF NOT FOUND, DEFAULT TO THE FIRST RELATIVE MINIMUM
C
        DO 500 I=FRMAX,N
            IF(MEXT(I).EQ.1 .AND. WMEAN(I).LT.0) THEN
                BITL(4) = I
                GOTO 510
            ENDIF
500    CONTINUE
        BITL(4) = FRMIN
510    CONTINUE
C
C  THE FIFTH ESTIMATE IS THE FIRST ZERO CROSSING AFTER THE FIRST RELATIVE
C  MAXIMUM OF THE MEAN
C
        BITL(5) = ZC
C
C  ADD THE DELAY DIFFERENCE
C

```

```

      DO 800 I=1,10
800      BITL(I) = BITL(I) + DELAY - 1

      RETURN
      END

C
C
C
C
C THIS SUBROUTINE IS TO DETERMINE IF A MAXIMUM OF THE VARIANCE
C IS CLOSE TO THE MINIMUM OF OF THE WEIGHT MEAN.
C
C The minimum of the mean is located at weight number I. If the
C maximum occurs within Q samples of I KEY is returned with the
C value of TRUE, otherwise it is returned with a value of FALSE.
C

      SUBROUTINE FMAX(X,I,Q,N,KEY)

      IMPLICIT NONE
      INTEGER I,Q,N,J
      REAL X(0:N)
      LOGICAL KEY

      KEY = .FALSE.

      DO 100 J=I-Q,I+Q
        IF (J-1.GE.0 .AND. J+1.LE.N-1) THEN
          IF (X(J).GE.X(J-1) .AND. X(J).GE.X(J+1)) KEY = .TRUE.
        ENDIF
100      CONTINUE

      RETURN
      END

```

```

C*****
C
C      ZCROSS
C
C      VAX-11 FORTRAN SOURCE FILENAME:          ZC.FOR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING      KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER(S)
C      -----          -
C      0.0              MAY 6, 1985      MARC D. BRACK
C*****
C
C      CALLING SEQUENCE
C
C          CALL ZCROSS(SIG,N,ZEST,NEST)
C
C      PURPOSE
C
C          This routine forms an estimate of the baud rate using
C          zero crossing analysis.
C
C      ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
C
C          NONE
C
C      ROUTINE(S) ACCESSED OR CALLED BY EMBEDDED ROUTINE(S)
C
C          NONE
C
C      ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C          SIG      Array of length N of input sequence (real)
C
C          N        Number of samples of input (integer)
C
C          NEST      Number of estimates to make; the different
C                   estimates are based on different criteria.
C                   NEST can be either: 1, 2, or 3. Generally,
C                   all estimates give same result.
C
C      ARGUMENT(S) SUPPLIED TO THE CALLING ROUTINE
C
C          ZEST      Array of length NEST of estimates (real)
C*****
C
C      SUBROUTINE ZCROSS(SIG,N,ZEST,NEST)
C
C      IMPLICIT NONE
C      INTEGER I,N,DIFF(2048),HIST(2048),J,K,SUCCESS,FAIL,NUM,NUM2,EST

```

```

      INTEGER MAXW,NEST
      REAL SIG(N),ZEST(NEST),DIFFA(2,2048),AMIN,TEMP,NORM,AMINT
      REAL RATIO(10),TOL(10),LMAXW

      DO 100 I=1,N
        DIFF(I) = 0
        HIST(I) = 0
100    CONTINUE

      I = 1
      J = 0
      K = 1

C
C  DETERMINE THE LOCATIONS OF THE ZERO CROSSINGS AND RECORD THE
C  DIFFERENCES IN SAMPLES BETWEEN THE CROSSINGS
C
      DO 210 WHILE(SIG(I)*SIG(I+1).GT.0)
        I=I+1
        IF(I.GT.N-1) GOTO 220
210    CONTINUE

215    IF (SIG(I+1).EQ.0) THEN
      I=I+1
      IF(I.GT.N-1) GOTO 220
      GOTO 215
    ELSE
      I=I+1
    ENDIF

    IF (I.GT.N-1) GOTO 220

    J=J+1
    DIFF(J) = I - K
    K = I
    GOTO 200

220    CONTINUE
C
C  MAKE A HISTOGRAM OF THE DIFFERENCESμ THE INDEX OF THE HIST ARRAY IS
C  THE DIFFERENCE BETWEEN CROSSINGS, THE VALUE OF HIST FOR A CERTAIN
C  INDEX IS THE NUMBER OF TIMES A DIFFERENCE WHICH EQUALS TO THE INDEX
C  OCCURRED
C
      DO 300 I=1,J
300    HIST(DIFF(I)) = HIST(DIFF(I)) + 1
C
C  MAKE IMPULSE FUNCTION ON REAL LINEμ THE LOCATION OF THE IMPULSES ARE
C  THE AVERAGE OF A GROUP IN THE HISTOGRAM WHERE A GROUP IS A COLLECITON
C  OF ENTRIES NOT HAVING TWO OR MORE ADJACENT EMPTY BINS. THE WEIGHTS
C  OF THE IMPULSES ARE THE NUMBER OF HISTOGRAM ENTRIES THAT CONTRIBUTED
C  TO THE AVERAGE OF THE RESPECTIVE IMPULSE. THE ARRAY DIFFA CONTAINS
C  THE LOCATION OF THE IMPULSES IN THE FIRST ROW AND THE WEIGHT IN THE

```

```

C SECOND ROW
C
      J=1
      NUM2 = 0
      DO 320 I=1,N
C
C TO DETECT A GROUP, FIRST FIND A NON-EMPTY BIN THAT HAS TWO EMPTY BINS
C IMMEDIATELY PRECEDING IT (CARE MUST BE TAKEN IN CHECKING THE FIRST
C TWO BINS). THEN GATHER IN BIN ENTRIES FROM THE FIRST BIN OF THE GROUP
C UNTIL TWO ADJACENT EMPTY BINS ARE FOUND
C
      IF(I.LE.2) THEN
        IF(HIST(1).NE.0 .AND. I.EQ.1) THEN
          NUM = HIST(I)
          DIFFA(1,J) = HIST(I)*I
          K=1
310      IF(.NOT.(HIST(I+K).EQ.0 .AND. HIST(I+K+1).EQ.0)) THEN
            NUM = NUM + HIST(I+K)
            DIFFA(1,J) = DIFFA(1,J) + HIST(I+K)*(I+K)
            K=K+1
            GOTO 310
          ENDIF
          DIFFA(1,J) = DIFFA(1,J)/NUM
          DIFFA(2,J) = NUM
          NUM2 = NUM2 + NUM
          J=J+1
        ENDIF
        IF(HIST(2).NE.0 .AND. HIST(1).EQ.0 .AND. I.EQ.2) THEN
          NUM = HIST(I)
          DIFFA(1,J) = HIST(I)*I
          K=1
312      IF(.NOT.(HIST(I+K).EQ.0 .AND. HIST(I+K+1).EQ.0)) THEN
            NUM = NUM + HIST(I+K)
            DIFFA(1,J) = DIFFA(1,J) + HIST(I+K)*(I+K)
            K=K+1
            GOTO 312
          ENDIF
          DIFFA(1,J) = DIFFA(1,J)/NUM
          DIFFA(2,J) = NUM
          NUM2 = NUM2 + NUM
          J=J+1
        ENDIF
      ELSE
        IF(HIST(I).NE.0 .AND. HIST(I-1).EQ.0 .AND. HIST(I-2).EQ.0) THEN
          NUM = HIST(I)
          DIFFA(1,J) = HIST(I)*I
          K=1
315      IF(.NOT.(HIST(I+K).EQ.0 .AND. HIST(I+K+1).EQ.0)) THEN
            NUM = NUM + HIST(I+K)
            DIFFA(1,J) = DIFFA(1,J) + HIST(I+K)*(I+K)
            K=K+1
            GOTO 315

```

```

        ENDIF
        DIFFA(1,J) = DIFFA(1,J)/NUM
        DIFFA(2,J) = NUM
        NUM2 = NUM2 + NUM
        J=J+1
    ENDIF
ENDIF
320    CONTINUE

    J=J-1
    AMIN = N
    DO 330 I=1,J
330      AMIN = MIN(AMIN,DIFFA(1,I))

    TOL(1) = 0.05
    TOL(2) = 0.20
    TOL(3) = 0.10
    RATIO(1) = 2
    RATIO(2) = 2
    RATIO(3) = 2

    DO 390 EST=1,NEST
C
C   DETERMINE THE NUMBER OF SUCCESS AND FAILURES OF MEETING THE TOLERANCE
C   CRITERION WHERE WHERE A SUCCESS IS WHEN THE LOCATION OF THE IMPULSE
C   CONSIDERED IS AN INTEGRAL DIVISOR OF THE A LOCATION TO THE RIGHT.
C   EACH IMPULSE TO THE RIGHT THAT SATASFIES THE CRITERION CONTRIBUTES ITS
C   WEIGHT TO THE TOTAL NUMBER OF SUCCESS
C
335      SUCCES = 0
      FAIL = 0
      DO 340 I=1,J
        IF(DIFFA(1,I).GE.AMIN) THEN
          TEMP = DIFFA(1,I)/AMIN
          NORM = (TEMP - ANINT(TEMP))/ANINT(TEMP)
          IF(ABS(NORM).LE.TOL(EST)) THEN
            SUCCES = SUCCES + DIFFA(2,I)
          ELSE
            FAIL = FAIL + DIFFA(2,I)
          ENDIF
          IF(DIFFA(1,I).EQ.AMIN) NUM = DIFFA(2,I)
        ENDIF
340      CONTINUE
C
C   CHECK THE RATIO CRITERION. THERE MUST BE A CERTAIN RATIO BETWEEN THE
C   NUMBER OF SUCCESS AND NUMBER OF FAILURES. IN ADDITION THE NUMBER OF
C   SUCCESS MUST BE GREATER THAN ONE-FOURTH OF THE TOTAL NUMBER OF
C   CROSSINGS AND THE NUMBER OF ENTRIES CONTRIBUTING TO THE IMPULSE
C   BEING CONSIDERED MUST BE GREATER THAN ONE-TENTH OF THE TOTAL NUMBER
C   OF CROSSINGS
C
      IF(SUCCES.GE.RATIO(EST)*FAIL .AND. SUCCES.GE.NUM2/4 .AND.

```

```

X      NUM.GT.NUM2/10) THEN
      ZEST(EST) = AMIN
      ELSE
        AMINT = AMIN
        AMIN = N
        DO 360 I=1,J
360      IF(MIN(DIFFA(1,I),AMIN).GT.AMINT) AMIN = MIN(DIFFA(1,I),AMIN)

        IF(AMIN.LT.N) THEN
          GOTO 335
        ELSE

C
C IF NO IMPULSE MEETS THE CRITERIA, CHOOSE THE MAXIMUM WEUGHT IMPULSE
C AS THE DEFAULT ESTIMATE
C
          MAXW = 0
          DO 370 I=1,J
            MAXW = MAX(DIFFA(2,I),FLOAT(MAXW))
370          CONTINUE
          DO 380 I=1,J
            IF (MAXW.EQ.DIFFA(2,I)) THEN
              LMAXW = DIFFA(1,I)
              GOTO 385
            ENDIF
380          CONTINUE
385          ZEST(EST) = LMAXW
          ENDIF
        ENDIF
390      CONTINUE

      RETURN
      END

```

C*****
C
C DLD
C
C VAX-11 FORTRAN SOURCE FILENAME: DLD.FOR
C
C DEPARTMENT OF ELECTRICAL ENGINEERING KANSAS STATE UNIVERSITY

```

C*****
C
C      DLD
C
C      VAX-11 FORTRAN SOURCE FILENAME:          DLD.FOR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING      KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER(S)
C      -----          -
C      0.0              JAN 23, 1983      DONALD M. HUMMELS
C      0.1              MAY 6, 1985      MARC D. BRACK
C
C*****
C
C      CALLING SEQUENCE
C
C      CALL DLD (TD,FC,FLAG,S,N,TF)
C
C      PURPOSE
C
C      This subroutine models a delay line discriminator.
C      No pre- or post filtering is done. The resulting
C      waveform is returned in the input array.
C
C      ROUTINE(S) ACCESSED OR CALLED BY THIS ROUTINE
C
C      DFT
C
C      ROUTINE(S) ACCESSED OR CALLED IN EMBEDDED ROUTINES
C
C      NONE
C
C      ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C      TD=Time delay of discriminator
C      FC=Center frequency
C      FLAG=Integer representing type of discriminator
C          0 -- Cos type
C          1 -- Sin type
C      S=Array containing input signal
C      N=# of points in input array (a power of 2)
C      TF=Input period (=1/fundamental freq)
C
C      ARGUMENT(S) SUPPLIED TO THE CALLING ROUTINE
C
C      S=Array containing output signal
C
C*****
C
SUBROUTINE DLD(TD,FC,FLAG,S,N,TF)

```

```

      COMPLEX S(2048),S2(2048),FACTOR
      INTEGER FLAG
C
C   PERFORM THE TIME DELAY
C
      DO 19 I = 1,N
      S2(I) = S(I)
19    CONTINUE
      CALL DFT(S2,N,0)

      DO 20 I = 1,N/2
      ANGLE = -(2*3.141592*FLOAT(I-1)/TF*TD)
      FACTOR = CMPLX(COS(ANGLE),SIN(ANGLE))
      S2(I) = S2(I) * FACTOR
      ANGLE = -(2*3.141592*FLOAT(-I)/TF*TD)
      FACTOR = CMPLX(COS(ANGLE),SIN(ANGLE))
      S2(N+1-I) = S2(N+1-I) * FACTOR
20    CONTINUE
      CALL DFT(S2,N,1)
C
C   FIND THE INPUT FOR THE POST-FILTER
C
      ANGLE = 3.14159*(2*FC*TD + .5*FLAG)
      FACTOR = CMPLX(COS(ANGLE),SIN(ANGLE))
      DO 40 I=1,N
      S(I) = S(I)*CONJG(S2(I))*FACTOR
      S(I) = CMPLX(4.*REAL(S(I)),0.)
40    CONTINUE

      RETURN
      END

```

References

1. B. Widrow, et al., 'Adaptive Noise Cancelling: Principles and Applications,' Proceedings of the IEEE, Vol. 63, No. 12, December 1975.
2. F. W. Ratcliffe, A Performance Analysis of the Delay Line Discriminator, Ph. D. Dissertation, Kansas State University, Manhattan, Kansas, 1984.
3. M. Schwartz, W. R. Bennett and S. Stein, Communication Systems and Techniques, McGraw-Hill Book Co., New York, New York, 1966.
4. D. R. Hnmels and F. W. Ratcliffe, 'Modeling Digital Communication Systems for Numerical Evaluation of Error Rate, Power Spectrum, Eye Pattern and Envelope,' Motorola Technical Report, Government Electronics Division, Scottsdale, Arizona, July 31, 1980.
5. J. A. Cadzow, 'Spectral Estimation: An Overdetermined Rational Model Equation Approach,' Proceedings of the IEEE, Vol. 70, No. 9, September 1982.

DETERMINATION OF THE BAUD RATE OF AN FSK SIGNAL
USING ADAPTIVE NOISE CANCELLING TECHNIQUES

by

MARC DAVID BRACK

B.S., Kansas State University, 1984

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1985

Abstract

The traditional method of FSK signal band rate estimation is to use a delay-line discriminator to demodulate the signal and then perform a zero-crossing analysis to obtain the band rate. This study proposes an alternate method which implements an adaptive noise canceller. The weight vector of the adaptive noise canceller, upon convergence, can be used directly to determine the band rate. However, the input signal is random making convergence difficult to obtain or sustain. The adaptive noise canceller can be used indirectly to improve the performance of the zero-crossing analysis. The way in which it does this is by making a rough but consistent estimate of the bandwidth of the message. This parameter is not known due to the surveillance situation the receiver is intended to be used in. Then the delay-line discriminator output can be filtered to produce a cleaner input for the zero-crossing analysis which will improve performance.

The performance of the traditional zero-crossing analysis and the direct and indirect implementation of the adaptive noise canceller was judged using a Monte Carlo simulation. The results showed that both adaptive noise cancelling methods performed better than the zero-crossing analysis alone.